DEALING WITH MISSING DATA IN EDUCATIONAL RESEARCH

SOFTWARE TUTORIALS

CRAIG K. ENDERS, PHD
MICHAEL P. WOLLER, MA

TABLE OF CONTENTS

DATA FILE DESCRIPTIONS

The analysis examples use synthetic data sets created to closely resemble those from educational studies described in Montague et al. (2005) and Montague et al. (2014). The data and analysis scripts are available for download from the project website: www.appliedmissingdata.com/videos.

The `behaviorachievement.dat` file is taken from a longitudinal study that followed 138 students from primary to middle school. The file includes three annual assessments of broad reading and math achievement beginning in the first grade, seventh grade standardized achievement test scores taken from a statewide assessment, and a final measure of broad reading and math obtained in ninth grade. The data also contain teacher ratings of behavioral symptoms and learning problems were also obtained in the first grade.

The `mathachievement.dat` data set is taken from an educational intervention where 250 students were assigned to an intervention and comparison condition. The file includes pretest and posttest math achievement scores, a measure of math self-efficacy, standardized reading scores taken from a statewide assessment, and several sociodemographic variables.

The `problemsolving2level.dat` data set is taken from a cluster-randomized educational intervention where 29 schools were assigned to an intervention and comparison condition. In addition to the intervention assignment indicator, school-level variables include the average years of teacher experience and the percentage of learners for whom English is a second language. The 928 student-level records include pretest and posttest math problem-solving and self-efficacy scores, standardized math scores taken from a statewide assessment, and several sociodemographic variables.

The `problemsolving3level.dat` data set is taken from a cluster-randomized educational intervention where 29 schools were assigned to an intervention and comparison condition. In addition to the intervention assignment indicator, school-level variables include the average years of teacher experience and the percentage of learners for whom English is a second language. The 928 student-level records include seven monthly measures of math problem-solving and self-efficacy, standardized math scores taken from a statewide assessment, and several sociodemographic variables.

Variable Definitions for behaviorachievement.dat File

Name	Definition	Missing %	Scale
ID	Individual identifier	0	Integer index
MALE	Gender dummy code	1.5	0 = Female, 1 = Male
HISPANIC	Hispanic dummy code	5.1	0 = African American, 1 = Hispanic
RISKGRP	Emotion/behavior disorder risk	2.2	1 = Low, 2 = Medium, 3 = High
ATRISK	Emotion/behavior disorder risk	2.2	0 = Low, 1 = Medium/high
BEHSYMP1	1st grade behavioral symptoms	3.6	Numeric (17 to 92)
LRNPROB1	1st grade learning problems	2.2	Numeric (31 to 88)
READ1	1st grade reading composite	6.5	Numeric (39 to 153)
READ2	2nd grade reading composite	9.4	Numeric (20 to 150)
READ3	3rd grade reading composite	14.5	Numeric (46 to 138)
READ9	9th grade reading composite	17.4	Numeric (41 to 123)
READ9GRP	9th grade reading classification	17.4	0 = Below average, 1 = Average
STANREAD7	7th grade standardized reading	19.6	Numeric (100 to 399)
MATH1	1st grade math composite	6.5	Numeric (60 to 149)
MATH2	2nd grade math composite	9.4	Numeric (76 to 138)
MATH3	3rd grade math composite	14.5	Numeric (71 to 143)
MATH9	9th grade math composite	17.4	Numeric (55 to 127)
MATHGRP9	9th grade math classification	17.4	0 = Below average, 1 = Average
STANMATH7	7th grade standardized math	19.6	Numeric (100 to 421)

Variable Definitions for mathachievement.dat File

Name	Definition	Missing %	Scale
ID	Individual identifier	0	Integer index
CONDITION	Experimental condition	0	0 = Comparison, 1 = Intervention
MALE	Gender dummy code	0	0 = Female, 1 = Male
FRLUNCH	Lunch assistance dummy code	4.4	0 = None, 1 = Lunch assistance
ATRISK	Emotion/behavior disorder risk	5.2	0 = Low risk, 1 = At-risk
STANREAD	Standardized reading	9.2	Numeric (27 to 69)
EFFICACY	Math self-efficacy rating scale	9.6	Ordinal (1 to 6)
ANXIETY	Math anxiety composite	8.4	Numeric (0 to 44)
MATHPRE	Math achievement pretest	0	Numeric (26 to 76)
MATHPOST	Math achievement posttest	18.0	Numeric (37 to 85)

Variable Definitions for problemsolving2level.dat File

Name	Definition	Missing %	Scale
SCHOOL	School identifier	0	Integer index
STUDENT	Student identifier	0	Integer index
CONDITION	Experimental condition	0	0 = Control, 1 = Experimental
TEACHEXP	Teacher years of experience	10.8	Numeric (4.3 to 24.6)
ESLPCT	% English as second language	0	Numeric (10 to 100)
ETHNIC	Ethnicity/race	9.0	1 = White, 2 = Black, 3 = Hispanic
MALE	Gender dummy code	0	0 = Female, 1 = Male
FRLUNCH	Lunch assistance code	4.7	0 = None, 1 = Lunch assistance
LOWACH	Low achievement code	5.2	0 = Typically achieving, 1 = Low achieving
STANMATH	Standardized math scores	7.4	Numeric (5.3 to 87.8)
EFFICACYPRE	Math self-efficacy pretest	0	Numeric (0 to 12)
EFFICACYPST	Math self-efficacy posttest	20.5	Numeric (0 to 12)
PSOLVEPRE	Math problem-solving pretest	0	Numeric (37 to 66)
PSOLVEPST	Math problem-solving posttest	20.5	Numeric (37 to 65)

Variable Definitions for problemsolving3level.dat File

Name	Definition	Missing %	Scale
SCHOOL	School identifier	0	Integer index
STUDENT	Student identifier	0	Integer index
WAVE	Monthly wave identifier	0	Integer index (1 to 7)
CONDITION	Experimental condition	0	0 = Control, 1 = Experimental
TEACHEXP	Teacher years of experience	10.8	Numeric (4.3 to 24.6)
ESLPCT	% English as second language	0	Numeric (10 to 100)
ETHNIC	Ethnicity/race	9.0	1 = White, 2 = Black, 3 = Hispanic
MALE	Gender dummy code	0	0 = Female, 1 = Male
FRLUNCH	Lunch assistance code	4.7	0 = None, 1 = Lunch assistance
LOWACH	Low achievement code	5.2	0 = Typically achieving, 1 = Low achieving
STANMATH	Standardized reading	7.4	Numeric (5.3 to 87.8)
MONTH0	Time scores (baseline = 0)	0	Numeric (0 to 6)
MONTH7	Time scores (endpoint = 0)	0	Numeric (–6 to 0)
EFFICACY	Math self-efficacy	11.4	Numeric (0 to 14)
PROBSOLVE	Math problem-solving	11.4	Numeric (37 to 68)

GETTING STARTED WITH SOFTWARE

The tutorial examples use the Blimp application for Bayesian estimation and multiple imputation. Blimp's development was supported by the Institute of Education Sciences, U.S. Department of Education, through Grant R305D150056 & R305D190002 to UCLA. Blimp is freely available at www.appliedmissingdata.com/blimp. The Blimp User Guide is available from the same website and from the `Help > Help` pull-down. To modify and customize features of the graphical interface (e.g., fonts, layout of the syntax and output panes, etc.), go to the `Blimp Studio > Preferences` pull-down.

The tutorial examples use Mplus for maximum likelihood estimation and for analyzing multiply imputed data sets created by Blimp. A free demo version of Mplus is available at www.statmodel.com/demo.shtml. Many of the scripts run on the demo version, which is limited to six variables.

The tutorial examples also use various R packages for maximum likelihood estimation and for analyzing multiply imputed data sets created by Blimp. R is available for download at www.r-project.org, and RStudio is available at www.rstudio.com/products/rstudio/download/. The script below installs the packages used in this document.

```
install.packages("lavaan", dependencies = T)
install.packages("semTools", dependencies = T)
install.packages("mitml", dependencies = T)
install.packages("mdmb", dependencies = T)
install.packages("remotes")
remotes::install_github("bkeller2/fdir")
```

SECTION 1: MAXIMUM LIKELIHOOD ESTIMATION

EXAMPLE 1: MULTIPLE REGRESSION WITH MULTIVARIATE NORMAL DATA

This example illustrates a multiple regression analysis with multivariate normal incomplete data. The analysis uses the `behaviorachievement.dat` data set taken from a longitudinal study that followed 138 students from primary through middle school. The file includes three annual assessments of broad reading and math achievement beginning in the first grade, seventh grade standardized achievement test scores taken from a statewide assessment, and a final measure of broad reading and math obtained in ninth grade. The data also contain teacher ratings of behavioral symptoms and learning problems were also obtained in the first grade. The data description at the beginning of this document provides additional details. The variables for this analysis are as follows.

Name	Definition	Missing %	Scale
BEHSYMP1	1st grade behavioral symptoms	3.6	Numeric
LRNPROB1	1st grade learning problems	2.2	Numeric
READ1	1st grade broad reading composite	6.5	Numeric
READ9	9th grade broad reading composite	17.4	Numeric

Analysis Model

The analysis model features ninth grade broad reading scores regressed on first grade reading achievement and teacher-rated learning problems and behavioral symptoms.

$$READ_9 = \beta_0 + \beta_1(READ_1) + \beta_2(LRNPROB_1) + \beta_3(BEHSYMP_1) + \varepsilon \quad (1)$$

Unlike a complete-data regression analysis, all incomplete variables require distributional assumptions, including the predictors. The Mplus and R scripts below assign a multivariate normal distribution to the set of analysis variables.

Mplus Script

The code block below shows Mplus script `Ex1.1.inp`.

Mplus Script Ex1.1.inp

```
1   DATA:
2   file = behaviorachievement.dat;
3   VARIABLE:
4   names = id male hispanic riskgrp atrisk behsymp1 lrnprob1
5     read1 read2 read3 read9 read9grp stanread7
6     math1 math2 math3 math9 math9grp stanmath7;
7   usevariables = read9 read1 lrnprob1 behsymp1;
8   missing = all(999);
9   ANALYSIS:
10  estimator = ml;
11  MODEL:
12  read1 lrnprob1 behsymp1;
13  read9 on read1 lrnprob1 behsymp1 (beta1-beta3);
14  MODEL TEST:
15  0 = beta1; 0 = beta2; 0 = beta3;
16  OUTPUT:
17  patterns sampstat stdyx cinterval;
```

The `DATA` command specifies the name of the input text file. No file path is required when the data set is located in the same directory as the script, as it is here. The `VARIABLE` command provides information about the data. Beginning on line 4, the `names` subcommand assigns names to the variables in the input data, the `usevariables` subcommand selects variables for the analysis, and the `missing` subcommand gives the global missing value code. The `ANALYSIS` command and `estimator` subcommand specify full information maximum likelihood estimation. These commands are optional because the maximum likelihood missing data handling is the default. If the variables are nonnormal, specifying `estimator = mlr` on line 10 generates robust test statistics and standard errors.

The `MODEL` section of the script consists of two lines. Listing all predictors by name on line 12 is important because doing so invokes a multivariate normal distribution for these variables. As mentioned previously, assigning distributional assumptions to predictors is necessary for missing

data handling. On line 13, the outcome variable appears to the left of the `on` keyword, and the predictors appear to the right. The end of this line includes labels for the slope parameters in parentheses. The subsequent `MODEL TEST` command uses these labels to specify a custom significance test of the omnibus null hypothesis that all three population slopes equal zero. Finally, the `OUTPUT` command specifies four keywords on line 17 that request a summary of the missing data patterns, maximum likelihood estimates of sample statistics, standardized coefficients, and confidence intervals.

Mplus Output

Information about the missing data patterns is found near the top of the output file. The table in the excerpt below shows the analysis variables in the rows and missing data patterns in the columns. The output also displays the frequency of each missing data pattern.

```
SUMMARY OF MISSING DATA PATTERNS


     MISSING DATA PATTERNS (x = not missing)

           1  2  3  4  5  6  7
 READ9     x  x  x  x
 READ1     x  x  x     x  x
 LRNPROB1  x  x     x  x  x  x
 BEHSYMP1  x     x  x  x     x


     MISSING DATA PATTERN FREQUENCIES

    Pattern   Frequency     Pattern   Frequency     Pattern   Frequency
          1          99           4           8           7           1
          2           4           5          22
          3           3           6           1
```

Next, the covariance coverage matrix displays the proportion of observed data for each variable on the diagonal and the proportion of observed data for each variable pair on the off-diagonals. A low value on the off-diagonal indicates that the data contain little information about a bivariate association.

```
COVARIANCE COVERAGE OF DATA

Minimum covariance coverage value   0.100

     PROPORTION OF DATA PRESENT

           Covariance Coverage
              READ9         READ1         LRNPROB1      BEHSYMP1
              --------      --------      --------      --------
 READ9          0.826
 READ1          0.768         0.935
 LRNPROB1       0.804         0.913         0.978
 BEHSYMP1       0.797         0.899         0.942         0.964
```

Most software programs that fit regression models report an omnibus *F* test that evaluates the set of slope coefficients. The `MODEL TEST` command in the previous script requested an analogous Wald chi-square statistic that evaluates the null hypothesis that all population slopes equal zero. The chi-square statistic, degrees of freedom, and *p*-value appear near the bottom of the `MODEL FIT INFORMATION` section under the `Wald Test of Parameter Constraints` heading. The test statistic is statistically significant, thus refuting the null hypothesis.

```
MODEL FIT INFORMATION

Number of Free Parameters                       14

...

Wald Test of Parameter Constraints

          Value                            159.666
          Degrees of Freedom                     3
          P-Value                           0.0000
```

The table of unstandardized parameter estimates is shown below. Because the analysis specifies a multivariate normal distribution for the predictors, the means, variances, and covariances of these variables are printed along with the focal model estimates. These supporting

parameters are not of substantive interest, and they do not need to be reported. The first two columns display the unstandardized estimates and their standard errors, and the third and fourth columns display the corresponding *z*-statistics and *p*-values. The focal model results are shown in bold typeface.

MODEL RESULTS

	Estimate	S.E.	Est./S.E.	Two-Tailed P-Value
READ9 ON				
READ1	**0.503**	**0.045**	**11.230**	**0.000**
LRNPROB1	**-0.224**	**0.132**	**-1.703**	**0.089**
BEHSYMP1	**-0.222**	**0.110**	**-2.023**	**0.043**
LRNPROB1 WITH				
READ1	-5.643	19.063	-0.296	0.767
BEHSYMP1 WITH				
READ1	-11.235	20.841	-0.539	0.590
LRNPROB1	92.048	13.548	6.794	0.000
Means				
READ1	86.732	1.709	50.739	0.000
LRNPROB1	52.328	0.914	57.224	0.000
BEHSYMP1	49.483	1.039	47.631	0.000
Intercepts				
READ9	**66.901**	**6.465**	**10.349**	**0.000**

```
Variances
    READ1            387.270     48.040      8.061      0.000
    LRNPROB1         114.162     13.820      8.260      0.000
    BEHSYMP1         146.318     17.738      8.249      0.000

 Residual Variances
    READ9             86.095     11.813      7.288      0.000
```

The results are interpreted in the same way as a complete-data regression analysis. For example, consider the first-grade reading slope. The model predicts that two individuals who differ by one point on `READ1` but are the same on `LRNPROB1` and `BEHSYMP1` should differ by .50 points on the outcome. The corresponding test statistic indicates that the slope coefficient is statistically different from zero ($z = 11.23$, $p < .001$).

Specifying the `stdyx` keyword as an option prints the table of standardized estimates and *R*-squared statistic shown below. The slope coefficients convey the expected change in standard deviation units for a one standard deviation increase in a given predictor. For example, the model predicts that two individuals who differ by one standard deviation on `READ1` but are the same on `LRNPROB1` and `BEHSYMP1` should differ by .68 standard deviations on the outcome. The *R*-squared statistic at the bottom of this section indicates that the collection predictors explain 59% of the variation in ninth-grade reading scores.

```
STANDARDIZED MODEL RESULTS

STDYX Standardization

                                                    Two-Tailed
                  Estimate       S.E.  Est./S.E.    P-Value

 READ9    ON
    READ1            0.683      0.049     13.901      0.000
    LRNPROB1        -0.165      0.097     -1.698      0.089
    BEHSYMP1        -0.185      0.091     -2.032      0.042

 LRNPROB1 WITH
    READ1            -0.027      0.091     -0.296      0.767
```

```
BEHSYMP1 WITH
    READ1             -0.047      0.087     -0.541      0.588
    LRNPROB1           0.712      0.042     16.784      0.000

 Means
    READ1              4.407      0.287     15.339      0.000
    LRNPROB1           4.897      0.309     15.864      0.000
    BEHSYMP1           4.091      0.262     15.594      0.000

 Intercepts
    READ9              4.620      0.575      8.032      0.000

 Variances
    READ1              1.000      0.000    999.000    999.000
    LRNPROB1           1.000      0.000    999.000    999.000
    BEHSYMP1           1.000      0.000    999.000    999.000

 Residual Variances
    READ9              0.411      0.059      6.974      0.000

R-SQUARE

    Observed                                        Two-Tailed
    Variable        Estimate       S.E.  Est./S.E.    P-Value

    READ9              0.589      0.059     10.014      0.000
```

R Script

The R input file for the analysis is `Ex1.1.R`. The code block below shows the commands that import the data.

R Script Ex1.1.R

```
1   library(fdir)
2   library(lavaan)
3   set()
4   data <- read.table("behaviorachievement.dat", na.strings = "999")
5   names(data) <-c("id","male","hispanic","riskgrp","atrisk","behsymp1",
6     "lrnprob1","read1","read2","read3","read9","read9grp","stanread7",
7     "math1","math2","math3","math9","math9grp","stanmath7")
```

The example requires the `fdir` and `lavaan` packages, which are loaded on lines 1 and 2. On line 3, the `set()` function of the `fdir` package identifies the file path of the folder containing the R script and sets this location as the working directory. On line 4, the `read.table` command imports the data, and the `na.strings` parameter specifies 999 as the global missing value code. It is only necessary to specify the name of the input data, as no file path is required when the file resides in the same folder as the R script, as it does here. Finally, variable names are listed beginning on line 5.

The code block below shows the `lavaan` syntax that fits the regression model and summarizes the results.

R Script Ex1.1.R, continued

```
 8  model <- 'read9 ~ b1*read1 + b2*lrnprob1 + b3*behsymp1'
 9  fit <- sem(model, data, fixed.x = F, missing = "fiml")
10  inspect(fit, "patterns")
11  inspect(fit, "coverage")
12  summary(fit, rsquare = T, standardize = T)
13  wald.constraints <- 'b1 == 0; b2 == 0; b3 == 0;'
14  lavTestWald(fit, constraints = wald.constraints)
```

The `model` variable on line 8 defines a text string specifying the regression model, with the outcome variable on the left side of the tilde and the predictors to the right. Each predictor's slope is preceded by a label (i.e., `b1`, `b2`, and `b3`). A subsequent command uses these labels to specify a custom significance test of the null hypothesis that the population slopes equal zero. On line 9, the model string and data frame are passed into the `sem` function. The `fixed.x = F` parameter specifies that the predictors are treated as normally distributed variables, and `missing = "fiml"` requests missing data estimation. The `fixed.x` specification is important because it invokes a multivariate normal distribution for the analysis variables. As mentioned previously, assigning distributions to incomplete predictors is necessary for missing data handling.

The `inspect` functions on lines 10 and 11 produce a table of missing data patterns and a covariance coverage matrix with the proportion of observed data for each variable or variable pair, respectively. The `summary` function on line 12 produces tabular results with standardized estimates and the *R*-squared statistic. Finally, the `wald.constraints` variable on line 13 defines a text string that uses the aforementioned labels to specify the null hypothesis that all three population slopes equal zero. The `lavTestWald` function on line 14 uses that text string to generate a chi-square statistic, degrees of freedom, and *p*-value.

R Output

The `inspect` functions in the previous script request information about the missing data patterns and missing data rates. The missing data pattern table in output below shows the analysis variables in the columns and missing data patterns in the rows (1 = observed, 0 = missing).

```
> inspect(fit, "patterns")
     read9 read1 lrnpr1 bhsym1
[1,]     1     1      1      1
[2,]     0     1      1      1
[3,]     1     0      1      1
[4,]     1     1      1      0
[5,]     1     1      0      1
[6,]     0     0      1      1
[7,]     0     1      1      0
```

The covariance coverage matrix displays the proportion of observed data for each variable on the diagonal and the proportion of observed data for each variable pair on the off-diagonals. A low value on the off-diagonal indicates that the data contain little information about a bivariate association.

```
> inspect(fit, "coverage")
         read9 read1 lrnpr1 bhsym1
read9    0.826
read1    0.768 0.935
lrnprob1 0.804 0.913 0.978
behsymp1 0.797 0.899 0.942  0.964
```

The table of parameter estimates is shown below. Because the analysis specifies a multivariate normal distribution for the predictors, the means, variances, and covariances of these variables are printed along with the focal model estimates. These supporting parameters are not of substantive interest, and they do not need to be reported. The first two columns display the unstandardized estimates and their standard errors, and the third and fourth columns display the corresponding *z*-statistics and *p*-values. The rightmost column gives the standardized coefficients. The focal model results are shown in bold typeface.

```
> summary(fit, rsquare = T, standardize = T)

...

Regressions:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
  read9 ~
    read1     (b1)    0.503    0.045   11.230    0.000    0.503    0.683
    lrnprob1  (b2)   -0.224    0.132   -1.702    0.089   -0.224   -0.165
    behsymp1  (b3)   -0.222    0.110   -2.023    0.043   -0.222   -0.185

Covariances:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
  read1 ~~
    lrnprob1         -5.637   19.063   -0.296    0.767   -5.637   -0.027
    behsymp1        -11.228   20.841   -0.539    0.590  -11.228   -0.047
  lrnprob1 ~~
```

```
    behsymp1          92.048   13.548    6.794    0.000   92.048    0.712


Intercepts:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
   .read9             66.901    6.465   10.349    0.000   66.901    4.620
    read1             86.732    1.709   50.739    0.000   86.732    4.407
    lrnprob1          52.328    0.914   57.225    0.000   52.328    4.897
    behsymp1          49.483    1.039   47.631    0.000   49.483    4.091

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
   .read9             86.096   11.813    7.288    0.000   86.096    0.411
    read1            387.275   48.041    8.061    0.000  387.275    1.000
    lrnprob1         114.160   13.820    8.260    0.000  114.160    1.000
    behsymp1         146.317   17.738    8.249    0.000  146.317    1.000

R-Square:
                   Estimate
    read9             0.589
```

The results are interpreted in the same way as a complete-data regression analysis. For example, consider the first-grade reading slope. The model predicts that two individuals who differ by one point on `READ1` but are the same on `LRNPROB1` and `BEHSYMP1` should differ by .50 points on the outcome. The corresponding test statistic indicates that the slope coefficient is statistically different from zero ($z = 11.23$, $p < .001$).

The standardized coefficients in the `Std.all` column convey the expected change in standard deviation units for a one standard deviation increase in a given predictor. For example, the model predicts that two individuals who differ by one standard deviation on `READ1` but are the same on `LRNPROB1` and `BEHSYMP1` should differ by .68 standard deviations on the outcome. The *R*-squared statistic at the bottom of this section indicates that the collection predictors explain 59% of the variation in ninth-grade reading scores.

Most software programs that fit regression models report an omnibus *F* test that evaluates the set of slope coefficients. The `lavTestWald` function in the previous script requested an analogous Wald chi-square statistic that evaluates the null hypothesis that all population slopes

equal zero. The chi-square statistic, degrees of freedom, and *p*-value appear on the output as follows. The test statistic is statistically significant, thus refuting the null hypothesis.

```
$stat
[1] 159.6636

$df
[1] 3

$p.value
[1] 0

$se
[1] "standard"
```

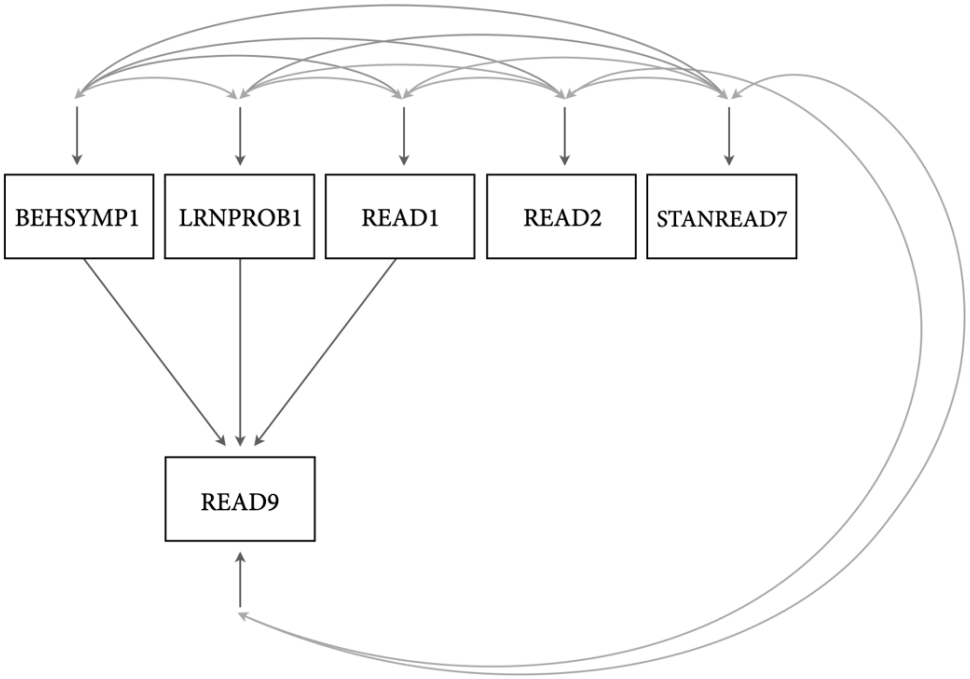
Adding Auxiliary Variables

The missing data literature often recommends an inclusive strategy that incorporates auxiliary variables that either predict missingness or correlate with the incomplete variables (Collins et al., 2001). The next part of example introduces two auxiliary variables using the saturated correlates approach described by Graham (2003). The analysis variables are as follows.

Name	Definition	Missing %	Scale
	Focal Variables		
BEHSYMP1	1st grade behavioral symptoms	3.6	Numeric
LRNPROB1	1st grade learning problems	2.2	Numeric
READ1	1st grade broad reading composite	6.5	Numeric
READ9	9th grade broad reading composite	17.4	Numeric
	Auxiliary Variables		
READ2	2nd grade broad reading composite	9.4	Numeric
STANREAD7	7th grade standardized math	19.6	Numeric

A path diagram of the saturated correlates model is shown below. The curved arrows depict correlations and residual correlations that connect the auxiliary variables to each other and to the residuals of the focal variables. Both Mplus and R have facilities that automatically introduce

auxiliary variables according to this model. Note that the saturated correlates approach assumes



that all variables are multivariate normal.

Mplus Script and Output

The code block below shows Mplus script `Ex1.2.inp`. The only change to the script is the `auxiliary` subcommand on line 8, which functions as a second variable list containing just the auxiliary variables. The `(m)` specification indicates that the additional variables are missing data auxiliary variables (Mplus uses this command for other purposes unrelated to missing data). Finally, note that the additional variables are omitted from the `usevariables` line.

Mplus Script Ex1.2.inp

```
DATA:
file = behaviorachievement.dat;
VARIABLE:
names = id male hispanic riskgrp atrisk behsymp1 lrnprob1
   read1 read2 read3 read9 read9grp stanread7
   math1 math2 math3 math9 math9grp stanmath7;
usevariables = read9 read1 lrnprob1 behsymp1;
auxiliary = (m) read2 stanread7;
missing = all(999);
ANALYSIS:
estimator = ml;
MODEL:
read1 lrnprob1 behsymp1;
read9 on read1 lrnprob1 behsymp1 (beta1-beta3);
MODEL TEST:
0 = beta1; 0 = beta2; 0 = beta3;
OUTPUT:
patterns sampstat stdyx cinterval;
```

The only indication that auxiliary variables are included in the model appears in the `SUMMARY OF ANALYSIS` table near the top of the output file. The main body of the output doesn't change with auxiliary variables, as the additional parameters (e.g., the curved arrows, or correlations) are suppressed. The estimates and standard errors may change, which is expected when including auxiliary variables that have salient semipartial correlations with the incomplete variables.

```
SUMMARY OF ANALYSIS

Number of groups                                         1
Number of observations                                 138

Number of dependent variables                            1
Number of independent variables                          3
Number of continuous latent variables                    0
```

```
Observed dependent variables

  Continuous
   READ9

Observed independent variables
   READ1       LRNPROB1    BEHSYMP1

Observed auxiliary variables
   READ2       STANREAD7
```

R Script and Output

The R input file that incorporates auxiliary variables is `Ex1.2.R`. In addition to the `fdir` and `lavaan` packages, the analysis requires the `semTools` package, which automates the inclusion of auxiliary variables. The code block below shows the syntax that fits the saturated correlates regression model. The `model` text string remains the same with auxiliary variables. The major change is that the `sem.auxiliary` function replaces the `sem` function from the earlier example, and the `aux` parameter defines a vector of auxiliary variable names. Unlike Mplus, the R output includes the auxiliary variable parameters. The additional estimates can be ignored because they are not the substantive focus.

```
1   model <- 'read9 ~ b1*read1 + b2*lrnprob1 + b3*behsymp1'
2   fit <- sem.auxiliary(model, data, fixed.x = F,
3     aux = c("read2","stanread7"))
```

EXAMPLE 2: LOGISTIC REGRESSION WITH A BINARY OUTCOME

This example illustrates a binary logistic regression analysis with incomplete data. The analysis uses the `behaviorachievement.dat` data set taken from a longitudinal study that followed 138 students from primary through middle school. The file includes three annual assessments of broad reading and math achievement beginning in the first grade, seventh grade standardized achievement test scores taken from a statewide assessment, and a final measure of broad reading and math obtained in ninth grade. The data also contain teacher ratings of behavioral symptoms and learning problems were also obtained in the first grade. The data description at the beginning of this document provides additional details. The variables for this analysis are as follows.

Name	Definition	Missing %	Scale
	Focal Variables		
BEHSYMP1	1st grade behavioral symptoms	3.6	Numeric
LRNPROB1	1st grade learning problems	2.2	Numeric
READ1	1st grade broad reading composite	6.5	Numeric
READ9GRP	9th grade reading classification	17.4	0 = Below average, 1 = Average
	Auxiliary Variables		
READ2	2nd grade broad reading composite	9.4	Numeric
STANREAD7	7th grade standardized math	19.6	Numeric

Analysis Model

The analysis model features a binary classification of ninth grade reading performance regressed on first grade reading achievement and teacher-rated learning problems and behavioral symptoms.

$$\text{logit}(READ9GRP) = \beta_0 + \beta_1(READ_1) + \beta_2(LRNPROB_1) + \beta_3(BEHSYMP_1) \qquad (2)$$

Unlike a complete-data regression analysis, all incomplete variables require distributional assumptions, including the predictors. Models with mixtures of categorical and numeric variables require a factored regression specification that separates the likelihood function into

separate components for each variable type. Mplus assigns a multivariate normal distribution to the predictors, whereas the R script links predictors to one another using a sequence of univariate regression models.

Mplus Script

The code block below shows Mplus script `Ex2.1.inp`.

Mplus Script Ex2.1.inp

```
1   DATA:
2   file = behaviorachievement.dat;
3   VARIABLE:
4   names = id male hispanic riskgrp atrisk behsymp1 lrnprob1
5     read1 read2 read3 read9 read9grp stanread7
6     math1 math2 math3 math9 math9grp stanmath7;
7   usevariables = read9grp read1 lrnprob1 behsymp1;
8   categorical = read9grp;
9   missing = all(999);
10  ANALYSIS:
11  estimator = ml;
12  link = logit;
13  integration = montecarlo;
14  MODEL:
15  read1 lrnprob1 behsymp1;
16  read9grp on read1 lrnprob1 behsymp1 (beta1-beta3);
17  MODEL TEST:
18  0 = beta1; 0 = beta2; 0 = beta3;
19  OUTPUT:
20  patterns sampstat stdyx cinterval;
```

The `DATA` command specifies the name of the input text file. No file path is required when the data file is located in the same directory as the script, as it is here. The `VARIABLE` command provides information about the data. Beginning on line 4, the `names` subcommand assigns names to the variables in the input data file, the `usevariables` subcommand selects variables for the analysis, and the `missing` subcommand gives the global missing value code. The `categorical` subcommand on line 8 defines the outcome as a binary variable. The `ANALYSIS` command and

`estimator` subcommand specify full information maximum likelihood estimation. Additionally, the `link = logit` subcommand specifies a logistic regression for the outcome variable, and `integration = montecarlo` invokes an algorithmic method for models with mixed variable types.

The `MODEL` section of the script consists of two lines. Listing all predictors by name on line 15 is important because doing so invokes a multivariate normal distribution for these variables. As mentioned previously, assigning distributional assumptions to predictors is necessary for missing data handling. On line 16, the outcome variable appears to the left of the `on` keyword, and the predictors appear to the right. The end of this line includes labels for the slope parameters in parentheses. The subsequent `MODEL TEST` command uses these labels to specify a custom significance test of the omnibus null hypothesis that all three population slopes equal zero. Finally, the `OUTPUT` command specifies four keywords on line 20 that request a summary of the missing data patterns, maximum likelihood estimates of sample statistics, standardized coefficients, and confidence intervals.

Mplus Output

Information about the missing data patterns is found near the top of the output file. Following the missing data pattern table, the output displays a covariance coverage matrix that gives the proportion of observed data for each variable on the diagonal and the proportion of observed data for each variable pair on the off-diagonals. The format of these table is the same as those shown in Example 1. In the interest of space, we point readers to that example for additional details.

Most software programs that fit regression models report an omnibus F test that evaluates the set of slope coefficients. The `MODEL TEST` command in the previous script requested an analogous Wald chi-square statistic that evaluates the null hypothesis that all population slopes equal zero. The chi-square statistic, degrees of freedom, and p-value appear near the bottom of the `MODEL FIT INFORMATION` section under the `Wald Test of Parameter Constraints` heading. The test statistic is statistically significant, thus refuting the null hypothesis.

```
MODEL FIT INFORMATION

Number of Free Parameters                       13

...

Wald Test of Parameter Constraints

          Value                             21.889
          Degrees of Freedom                    3
          P-Value                           0.0001
```

The table of unstandardized parameter estimates is shown below. Because the analysis specifies a multivariate normal distribution for the predictors, the means, variances, and covariances of these variables are printed along with the focal model estimates. These supporting parameters are not of substantive interest, and they do not need to be reported. The first two columns display the unstandardized estimates and their standard errors, and the third and fourth columns display the corresponding *z*-statistics and *p*-values. The focal model results are shown in bold typeface.

```
MODEL RESULTS

                                                    Two-Tailed
                  Estimate       S.E.  Est./S.E.    P-Value

 READGRP9   ON
    READ1            0.069      0.016      4.446      0.000
    LRNPROB1        -0.018      0.033     -0.549      0.583
    BEHSYMP1        -0.028      0.028     -1.014      0.311

 LRNPROB1 WITH
    READ1            3.085     19.553      0.158      0.875
```

```
BEHSYMP1 WITH
    READ1             -5.194     21.046     -0.247      0.805
    LRNPROB1          92.088     13.554      6.794      0.000

 Means
    READ1             86.974      1.719     50.598      0.000
    LRNPROB1          52.319      0.914     57.267      0.000
    BEHSYMP1          49.488      1.041     47.544      0.000

 Thresholds
    READGRP9$1         3.874      1.729      2.240      0.025

 Variances
    READ1            384.526     47.859      8.035      0.000
    LRNPROB1         113.906     13.775      8.269      0.000
    BEHSYMP1         146.740     17.818      8.235      0.000
```

The results are interpreted in the same way as a complete-data logistic regression analysis. For example, consider the first-grade reading score slope. The model predicts that the logits for two individuals who differ by one point on `READ1` but are the same on `LRNPROB1` and `BEHSYMP1` differ by 0.07. The corresponding test statistic indicates that the slope coefficient is statistically different from zero ($z = 4.45$, $p < .001$). Note that Mplus reports a threshold parameter instead of the usual regression intercept. The threshold from a binary logistic model has the same value as the intercept but the opposite sign (i.e., $\hat{\beta}_0 = -3.87$).

Finally, the printed output also includes the table of odds ratios that reflect multiplicative changes to the odds. For example, a one-point increase in first grade reading scores increases the odds of achieving an average ninth grade reading level by a factor 1.07, holding first grade learning problems and behavioral symptoms constant.

```
LOGISTIC REGRESSION ODDS RATIO RESULTS

                                                 95% C.I.
                    Estimate       S.E.  Lower 2.5% Upper 2.5%

 READGRP9   ON
    READ1              1.072      0.017      1.040      1.105
    LRNPROB1           0.982      0.032      0.921      1.047
    BEHSYMP1           0.972      0.027      0.921      1.027
```

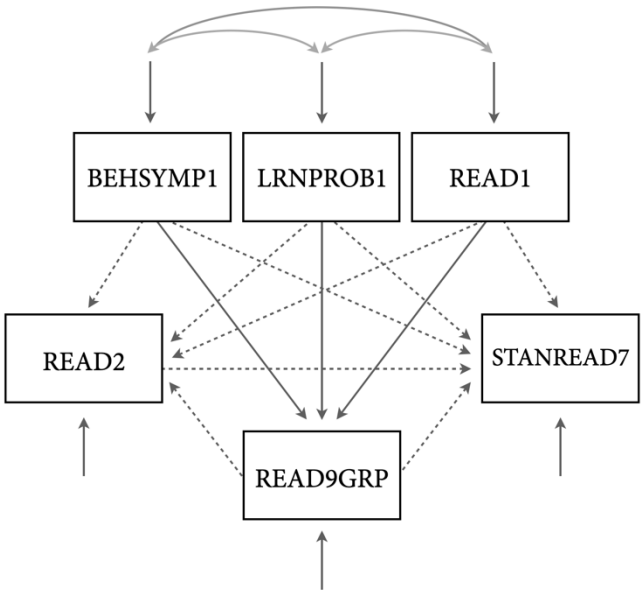
Adding Auxiliary Variables

The missing data literature often recommends an inclusive strategy that incorporates auxiliary variables that either predict missingness or correlate with the incomplete variables (Collins et al., 2001). The saturated correlates model from Example 1 is not applicable to logistic regression models because it assumes multivariate normality. Instead, auxiliary variables enter the model as additional outcomes that are predicted by the analysis variables and by each other.

The additional regression equations are as follows.

$$\begin{aligned} READ_2 &= \gamma_{01} + \gamma_{11}(READ9GRP) + \gamma_{21}(READ_1) \\ &+ \gamma_{31}(LRNPROB_1) + \gamma_{41}(BEHSYMP_1) + \epsilon_1 \\ STANREAD_7 &= \gamma_{02} + \gamma_{12}(READ_2) + \gamma_{22}(READ9GRP) \\ &+ \gamma_{32}(READ_1) + \gamma_{42}(LRNPROB_1) + \gamma_{52}(BEHSYMP_1) + \epsilon_2 \end{aligned} \tag{3}$$

Along with the logistic regression model from Equation 2, the collection of regression equations can be viewed as the path model shown below, where the dashed lines are the additional regressions. With this method, the focal model is one part of a larger network of variables. Importantly, the path model does not represent substantive theory, but is simply a tool for linking the auxiliary variables to the focal variables and to each other.



Mplus Script

The code block below shows an excerpt from Mplus script `Ex2.2.inp`. The `MODEL` command includes two new regression equations, but the script is otherwise similar to `Ex2.1.inp`.

```
MODEL:
read1 lrnprob1 behsymp1;
read9grp on read1 lrnprob1 behsymp1 (beta1-beta3);
read2 on read9grp read1 lrnprob1 behsymp1;
stanread7 on read2 read9grp read1 lrnprob1 behsymp1;
```

The main table of results expands to include summaries of the auxiliary variable regression models. However, these additional parameters can be ignored because they are not the substantive focus. The logistic model's estimates and standard errors change, which is expected when including auxiliary variables that have salient semipartial correlations with the incomplete variables.

R Script

The `lavaan` package currently does not offer maximum likelihood estimation for models with incomplete categorical variables. The example instead uses the `mdmb` package. This package

leverages a factored regression specification that links incomplete predictors to one another using a sequence of univariate regression models. The additional regression equations are as follows.

$$\begin{aligned} BEHSYMP_1 &= \gamma_{01} + \epsilon_1 \\ LRNPROB_1 &= \gamma_{02} + \gamma_{12}(BEHSYMP_1) + \epsilon_2 \\ READ_1 &= \gamma_{03} + \gamma_{13}(LRNPROB_1) + \gamma_{23}(BEHSYMP_1) + \epsilon_3 \end{aligned} \tag{4}$$

These equations essentially comprise a path model where first grade behavioral symptom ratings predict learning problems, and both variables then predict first grade reading scores.

The R input file for the analysis is `Ex2.R`. The code block below shows the commands that import the data.

R Script Ex2.R

```
1   library(fdir)
2   library(mdmb)
3   set()
4   data <- read.table("behaviorachievement.dat", na.strings = "999")
5   names(data) <-c("id","male","hispanic","riskgrp","atrisk","behsymp1",
6     "lrnprob1","read1","read2","read3","read9","read9grp","stanread7",
7     "math1","math2","math3","math9","math9grp","stanmath7")
8   summary(data[,c("stanread7","read2","read1","lrnprob1","behsymp1")])
```

The example requires the `fdir` and `mdmb` packages, which are loaded on lines 1 and 2. On line 3, the `set()` function of the `fdir` package identifies the file path of the folder containing the R script and sets this location as the working directory. On line 4, the `read.table` command imports the data, and the `na.strings` parameter specifies 999 as the global missing value code. It is only necessary to specify the name of the input data, as no file path is required when the file resides in the same folder as the R script, as it does here. Finally, variable names are listed beginning on line 5.

The `mdmb` package requires the user to specify "nodes" for the missing values. These nodes are essentially a fixed list of plausible score values that span each variable's range. Specifying these

values is necessary for the optimization algorithm, which uses an imputation-like algorithm called numerical integration. The `summary` function on line 8 generates a table displaying the observed values of the numeric variables. The summary table is as follows.

```
   stanread7         read2            read1           lrnprob1        behsymp1
Min.   :100.0   Min.   : 20.00   Min.   : 39.00   Min.   :31.00   Min.   :17.00
1st Qu.:228.0   1st Qu.: 83.00   1st Qu.: 74.00   1st Qu.:45.00   1st Qu.:41.00
Median :263.0   Median : 92.00   Median : 86.00   Median :51.00   Median :48.00
Mean   :264.5   Mean   : 93.74   Mean   : 86.81   Mean   :52.36   Mean   :49.47
3rd Qu.:314.0   3rd Qu.:108.00   3rd Qu.: 99.00   3rd Qu.:60.50   3rd Qu.:58.00
Max.   :399.0   Max.   :150.00   Max.   :153.00   Max.   :88.00   Max.   :92.00
NA's   :27      NA's   :13       NA's   :9        NA's   :3       NA's   :5
```

The next part of the code creates variables that contain vectors of plausible replacement scores (nodes, pseudo-imputations) that span the entire range of the distributions. The binary outcome has only two possible scores, so its node vector on line 12 consists of 0s and 1s. For continuous variables, specifying 20 to 40 nodes is usually sufficient. For example, `nodes.read1` is a vector of plausible scores ranging from 30 to 160 in increments of two, and `nodes.lrnprb1` is a sequence of scores between 20 and 100 in increments of two. To account for the possibility that the missing scores fall outside the observed range, the vectors specify values beyond the minimum and maximum scores from the data.

R Script Ex2.R, continued

```
10  nodes.stanread7 <- seq(80, 420, by = 5)
11  nodes.read2 <- seq(10, 160, by = 2)
12  nodes.read9grp <- c(0,1)
13  nodes.read1 <- seq(30, 160, by = 2)
14  nodes.lrnprob1 <- seq(20, 100, by = 2)
15  nodes.behsymp1 <- seq(10, 100, by = 2)
```

The next part of the script specifies a model for each analysis variable and auxiliary variable. The predictor variable regressions from Equation 4 are listed first, followed by the logistic model from Equation 2. The auxiliary variable regressions from Equation 3 are last. Each model object includes three arguments: the type of regression (linear or logistic), an equation, and the

incomplete variable's vector of nodes or pseudo-imputations. Linear regressions are specified with `"model" = "linreg"` parameter, and the binary logistic regression is specified using `"model" = "logistic"`.

R Script Ex2.R, continued

```
16  model.behsymp1 <- list("model" = "linreg",
17    "formula" = behsymp1 ~ 1, nodes = nodes.behsymp1)
18  model.lrnprob1 <- list("model" = "linreg",
19    "formula" = lrnprob1 ~ behsymp1, nodes = nodes.lrnprob1)
20  model.read1 <- list("model" = "linreg",
21     "formula" = read1 ~ lrnprob1 + behsymp1, nodes = nodes.read1)
22  model.read9grp <- list("model" = "logistic",
23     "formula" = read9grp ~ read1 + lrnprob1 + behsymp1,
24     nodes = nodes.read9grp)
25  model.read2 <- list("model" = "linreg",
26    "formula" = read2 ~ read9grp + read1 + lrnprob1 + behsymp1,
27    nodes = nodes.read2)
28  model.stanread7 <- list("model" = "linreg",
29    "formula" = stanread7 ~ read2 + read9grp + read1 + lrnprob1 +
30    behsymp1, nodes = nodes.stanread7)
```

The `mdmb` package views `stanread7` (the auxiliary variable in the final regression model) as the ultimate "dependent" variable in the sequence, and it considers all other variables "independent variables". Starting on line 31, the final part of the code combines the independent variable models into a list. On line 34, the data frame and the predictor list are passed into the `frm_em` function, which fits the sequence of models. Finally, the `summary` function on line 36 requests tables of parameter estimates.

R Script Ex2.R, continued

```
31  predictor.models <- list(behsymp1 = model.behsymp1, lrnprob1 =
32    model.lrnprob1, read1 = model.read1, read9grp = model.read9grp,
33    read2 = model.read2)
34  fit <- frm_em(dat = data, dep = model.stanread7, ind =
35    predictor.models)
```

```
36   summary(fit)
```

R Output

The `mdmb` output includes a table of results for every fitted regression model. In this example, the output tables summarize linear regressions for the three incomplete predictors, a logistic regression for the binary dependent variable, and a pair of linear regressions for the auxiliary variables. These supporting model parameters are not of substantive interest, and they do not need to be reported. The output below shows the parameter estimates from the focal logistic model. The first two columns display the unstandardized estimates and their standard errors, the third and fourth columns display the corresponding *t*-statistics and *p*-values, and the rightmost columns contain 95% confidence interval limits.

```
Model 4: mdmb::logistic_regression( read9grp ~ read1 + lrnprob1 + behsymp1 )

  index       dv                    parm ON     est     se       t      p lower95 upper95
1    14 read9grp read9grp ON (Intercept)  1 -3.9045 1.6291 -2.3968 0.0165 -7.0974 -0.7117
2    15 read9grp       read9grp ON read1  1  0.0675 0.0149  4.5252 0.0000  0.0383  0.0968
3    16 read9grp    read9grp ON lrnprob1  1 -0.0225 0.0308 -0.7330 0.4636 -0.0828  0.0377
4    17 read9grp    read9grp ON behsymp1  1 -0.0192 0.0251 -0.7664 0.4434 -0.0685  0.0300

Pseudo R^2 (McKelvey & Zavoina)=0.4944
```

The results are interpreted in the same way as a complete-data logistic regression analysis. For example, consider the first-grade reading score slope. The model predicts that the logits for two individuals who differ by one point on `READ1` but are the same on `LRNPROB1` and `BEHSYMP1` differ by 0.07. The corresponding test statistic indicates that the slope coefficient is statistically different from zero ($t = 4.53$, $p < .001$).

EXAMPLE 3: REGRESSION WITH BINARY AND ORDINAL PREDICTORS

This example illustrates a multiple regression analysis with incomplete categorical predictors. The analysis uses the `mathachievement.dat` data set taken from an educational intervention where 250 students were assigned to an intervention and comparison condition. The file includes pretest and posttest math achievement scores, a measure of math self-efficacy, standardized reading scores taken from a statewide assessment, and several sociodemographic variables. The analysis variables are as follows.

Name	Definition	Missing %	Scale
	Focal Variables		
MATHPOST	Math achievement posttest	18.0	Numeric
CONDITION	Experimental condition	0	0 = Comparison, 1 = Intervention
FRLUNCH	Lunch assistance code	4.4	0 = None, 1 = Free/reduced lunch
EFFICACY	Math self-efficacy rating	9.6	Ordinal (1 to 6)
MATHPRE	Math achievement pretest	0	Numeric
	Auxiliary Variables		
ATRISK	Behavioral disorder risk	5.2	0 = Low risk, 1 = At-risk
STANREAD	Standardized reading	9.2	Numeric

Analysis Model

The analysis model features math posttest scores regressed on the experimental condition and lunch assistance dummy codes, math self-efficacy ratings, and math pretest scores.

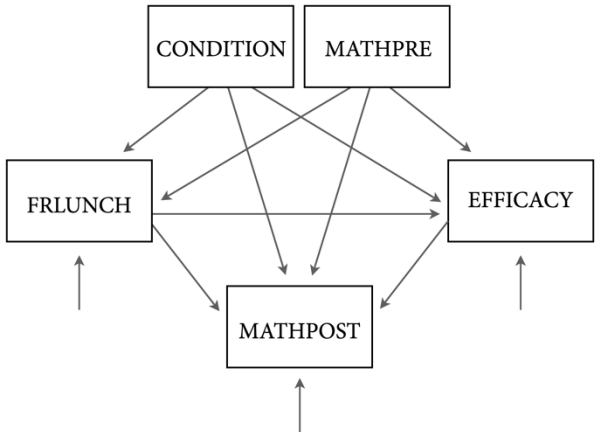
$$MATHPOST = \beta_0 + \beta_1(CONDITION) + \beta_2(FRLUNCH) + \beta_3(EFFICACY) + \beta_4(MATHPRE) + \varepsilon \quad (5)$$

Unlike a complete-data regression analysis, all incomplete variables require distributional assumptions, including the predictors. In this case, the predictor set includes incomplete binary and ordinal variables, so assigning a normal distribution to the variables is questionable.

The analysis instead uses a factored regression specification that separates the likelihood function into separate components for each variable type. In practical terms, this specification uses a sequence of univariate regression models to link incomplete predictors. The additional regression equations are both logistic regressions.

$$\begin{aligned} \text{logit}(FRLUNCH) &= \gamma_{01} + \gamma_{11}(CONDITION) + \gamma_{21}(MATHPRE) \\ \text{logit}(EFFICACY) &= \gamma_{02} + \gamma_{12}(FRLUNCH) + \gamma_{22}(CONDITION) + \gamma_{32}(MATHPRE) \end{aligned} \tag{6}$$

These equations comprise the path model below, where the intervention indicator and math pretest scores predict the lunch assistance indicator, and all three variables, in turn, predict self-efficacy. The two complete variables are always on the right side of regression equations because they do not require a model. The absence of residual arrows in the path diagram conveys this feature.



The missing data literature often recommends an inclusive strategy that incorporates auxiliary variables that either predict missingness or correlate with the incomplete variables (Collins et al., 2001). Following Example 2, auxiliary variables enter the model as additional outcomes that are predicted by the analysis variables and by each other. The additional regression equations are as follows.

$$\begin{aligned} \text{logit}(ATRISK) &= \gamma_{03} + \gamma_{13}(MATHPOST) + \gamma_{23}(CONDITION) \\ &+ \gamma_{33}(FRLUNCH) + \gamma_{43}(EFFICACY) + \gamma_{53}(MATHPRE) \\ STANREAD &= \gamma_{04} + \gamma_{14}(ATRISK) + \gamma_{24}(MATHPOST) + \gamma_{34}(CONDITION) \end{aligned} \tag{7}$$

$$+ \gamma_{44}(FRLUNCH) + \gamma_{54}(EFFICACY) + \gamma_{64}(MATHPRE) + \epsilon_4$$

Again, the entire collection of regression equations can be viewed as a path model (see the auxiliary variable path diagram from Example 2). The key difference is that the path coefficients are just a tool for linking variables with different metrics and do not represent a substantive theory.

Mplus Script

The code block below shows Mplus script `Ex3.inp`.

Mplus Script Ex3.inp

```
DATA:
file = mathachievement.dat;
VARIABLE:
names = id condition male frlunch atrisk
  stanread efficacy anxiety mathpre mathpost;
usevariables = mathpost condition frlunch efficacy
  mathpre atrisk stanread;
categorical = frlunch efficacy atrisk;
missing = all(999);
ANALYSIS:
estimator = ml;
link = logit;
integration = montecarlo;
MODEL:
frlunch on condition mathpre;
efficacy on frlunch condition mathpre;
mathpost on condition frlunch efficacy mathpre (beta1-beta4);
atrisk on mathpost condition frlunch efficacy mathpre;
stanread on atrisk mathpost condition frlunch efficacy mathpre;
MODEL TEST:
0 = beta1; 0 = beta2; 0 = beta3; 0 = beta4;
OUTPUT:
patterns sampstat cinterval;
```

The `DATA` command specifies the name of the input text file. No file path is required when the data file is located in the same directory as the script, as it is here. The `VARIABLE` command provides information about the data. Beginning on line 4, the `names` subcommand assigns names to the variables in the input data file, the `usevariables` subcommand selects variables for the analysis, and the `missing` subcommand gives the global missing value code. The `categorical` subcommand on line 8 defines three variables as either binary or ordinal. The `ANALYSIS` command and `estimator` subcommand specify full information maximum likelihood estimation. Finally, the `link = logit` option specifies a logistic regression for the outcome variable, and `integration = montecarlo` invokes an algorithmic method for models with mixed variable types (and a factored regression specification for the likelihood).

The `MODEL` section of the script consists of five lines. Lines 15 and 16 are logistic regressions linking the discrete predictors to the complete variables and each other (see Equation 6), and line 17 is the focal regression model from Equation 5. The end of this line includes parameter labels in parentheses. Finally, lines 18 and 19 are the auxiliary variable regressions shown in Equation 7. As noted previously, the collection of regressions can be viewed as a path model, with the focal regression as one part of the larger network of variables. Next, the `MODEL TEST` command uses the labels from line 17 to specify a custom significance test of the null hypothesis that all three population slopes equal 0. Finally, the `OUTPUT` command specifies three keywords on line 23 that request a summary of the missing data patterns, maximum likelihood estimates of sample statistics, and confidence intervals (standardized coefficients are not available for this analysis).

Mplus Output

Information about the missing data patterns is found near the top of the output file. Following the missing data pattern table, the output displays a covariance coverage matrix that gives the proportion of observed data for each variable on the diagonal and the proportion of observed data for each variable pair on the off-diagonals. The format of these table is the same as those shown in Example 1. In the interest of space, we point readers to that example for additional details.

Most software programs that fit regression models report an omnibus F test that evaluates the set of slope coefficients. The `MODEL TEST` command in the previous script requested an analogous Wald chi-square statistic that evaluates the null hypothesis that all population slopes equal zero. The chi-square statistic, degrees of freedom, and p-value appear near the bottom of the `MODEL FIT INFORMATION` section under the `Wald Test of Parameter Constraints` heading. The test statistic is statistically significant, thus refuting the null hypothesis.

```
MODEL FIT INFORMATION

Number of Free Parameters                       31

...

Wald Test of Parameter Constraints

          Value                            149.182
          Degrees of Freedom                     4
          P-Value                           0.0000
```

The table of unstandardized parameter estimates is shown below. Because the analysis specifies a multivariate normal distribution for the predictors, the means, variances, and covariances of these variables are printed along with the focal model estimates. These supporting parameters are not of substantive interest, and they do not need to be reported. The first two columns display the unstandardized estimates and their standard errors, and the third and fourth columns display the corresponding *z*-statistics and *p*-values. The focal model results are shown in bold typeface.

```
MODEL RESULTS

                                                    Two-Tailed
                  Estimate       S.E.  Est./S.E.    P-Value

 FRLUNCH    ON
    CONDITION          0.011      0.265      0.041      0.967
    MATHPRE           -0.020      0.015     -1.290      0.197

 EFFICACY   ON
    FRLUNCH           -0.031      0.246     -0.125      0.901
    CONDITION          0.506      0.240      2.107      0.035
    MATHPRE            0.056      0.014      3.881      0.000
```

MATHPOST ON				
CONDITION	**2.306**	**1.023**	**2.255**	**0.024**
FRLUNCH	**-5.498**	**1.063**	**-5.173**	**0.000**
EFFICACY	**0.833**	**0.340**	**2.448**	**0.014**
MATHPRE	**0.526**	**0.061**	**8.594**	**0.000**
ATRISK ON				
MATHPOST	-0.028	0.025	-1.141	0.254
CONDITION	-0.080	0.342	-0.233	0.815
FRLUNCH	0.898	0.399	2.248	0.025
EFFICACY	-0.337	0.115	-2.925	0.003
MATHPRE	-0.018	0.024	-0.722	0.470
STANREAD ON				
ATRISK	-13.492	1.231	-10.957	0.000
MATHPOST	0.349	0.078	4.466	0.000
CONDITION	1.493	1.019	1.466	0.143
FRLUNCH	-2.435	1.177	-2.068	0.039
EFFICACY	-0.478	0.351	-1.361	0.173
MATHPRE	0.006	0.073	0.076	0.939
Intercepts				
MATHPOST	**29.375**	**3.016**	**9.739**	**0.000**
STANREAD	44.135	4.035	10.938	0.000
Thresholds				
FRLUNCH$1	-0.623	0.780	-0.799	0.425
EFFICACY$1	1.308	0.748	1.748	0.080
EFFICACY$2	2.213	0.752	2.942	0.003
EFFICACY$3	3.250	0.770	4.222	0.000
EFFICACY$4	4.186	0.786	5.324	0.000
EFFICACY$5	4.976	0.800	6.217	0.000
ATRISK$1	-4.351	1.317	-3.304	0.001
Residual Variances				
MATHPOST	**51.270**	**5.185**	**9.888**	**0.000**
STANREAD	52.261	5.226	10.000	0.000

The results are interpreted in the same way as a complete-data regression analysis with categorical predictors. For example, consider the slope for the treatment assignment dummy code. The positive coefficient indicates that, for two students who share the same covariate profile (i.e., lunch assistance, self-efficacy, and pretest scores), the model predicts that the student in the experimental condition should score 2.31 points higher than the student in the control group. The corresponding test statistic indicates that the slope coefficient is statistically different from zero ($z = 2.26$, $p = .02$).

R Script

The `lavaan` package currently does not offer maximum likelihood estimation for models with incomplete categorical variables. The example instead uses the `mdmb` package. This package leverages the same factored regression specification described previously. The R input file for the analysis is `Ex3.R`. The code block below shows the commands that import the data.

R Script Ex3.R

```
1   library(fdir)
2   library(mdmb)
3   set()
4   data <- read.table("mathachievement.dat", na.strings = "999")
5   names(data) <- c("id", "condition","male","frlunch", "atrisk",
6     "stanread","efficacy", "anxiety", "mathpre", "mathpost")
7   summary(data[,c("mathpost","stanread")])
```

The example requires the `fdir` and `mdmb` packages, which are loaded on lines 1 and 2. On line 3, the `set()` function of the `fdir` package identifies the file path of the folder containing the R script and sets this location as the working directory. On line 4, the `read.table` command imports the data, and the `na.strings` parameter specifies 999 as the global missing value code. It is only necessary to specify the name of the input data file. No file path is required when the data reside in the same folder as the R script as is the case here. Finally, variable names are listed beginning on line 5.

The `mdmb` package requires the user to specify "nodes" or "pseudo-imputations" for the missing values. These nodes are essentially a fixed list of plausible score values that span each variable's range. Specifying these values is necessary for the optimization algorithm, which uses

an imputation-like algorithm called numerical integration. The `summary` function on line 7 generates a table displaying the observed values of the numeric variables. The discrete variables are excluded because their ranges are either 0 and 1 (the binary codes) or 1 to 6 (ordinal self-efficacy ratings). The summary table is as follows.

```
   mathpost        stanread
Min.   :37.00   Min.   :27.00
1st Qu.:52.00   1st Qu.:45.00
Median :57.00   Median :55.00
Mean   :57.45   Mean   :52.52
3rd Qu.:63.00   3rd Qu.:60.50
Max.   :85.00   Max.   :69.00
NA's   :45      NA's   :23
```

The next part of the code creates variables that contain vectors of plausible replacement scores (nodes, pseudo-imputations) that span the entire range of the distributions. The binary variables have only two possible scores, so their node vectors on lines 8 and 11 consist of 0s and 1s. On line 9, the efficacy scores similarly use integer nodes between 1 and 6. For continuous variables, specifying 20 to 40 nodes is usually sufficient. For example, `nodes.stanread` is a vector of plausible scores ranging from 20 to 80 in increments of two, and `nodes.mathpost` is a sequence of scores between 30 and 90 in increments of two. To account for the possibility that the missing scores fall outside the observed range, the vectors specify values beyond the minimum and maximum scores from the data.

R Script Ex3.1.R, continued

```
 8  nodes.frlunch <- c(0,1)
 9  nodes.efficacy <- seq(1, 6, by = 1)
10  nodes.mathpost <- seq(30, 90, by = 2)
11  nodes.atrisk <- c(0,1)
12  nodes.stanread <- c(20, 80, by = 2)
```

The next part of the script specifies a model for each analysis variable and auxiliary variable. The predictor variable regressions from Equation 6 are listed first, followed by the focal model from Equation 5. The auxiliary variable regressions from Equation 7 are last. Each model object includes three arguments: the type of regression (linear or logistic), an equation, and the incomplete variable's nodes. Linear regressions are specified with `"model" = "linreg"` parameter, and the binary logistic regression is specified using `"model" = "logistic"`.

R Script Ex3.1.R, continued

```
13  model.frlunch <- list("model" = "logistic",
14    "formula" = frlunch ~ condition + mathpre,
15    nodes = nodes.frlunch)
16  model.efficacy <- list("model" = "linreg",
17    "formula" = efficacy ~ frlunch + condition + mathpre,
18    nodes = nodes.efficacy)
19  model.mathpost <- list("model" = "linreg",
20    "formula" = mathpost ~ condition + frlunch + efficacy + mathpre,
21    nodes = nodes.mathpost)
22  model.atrisk <- list("model" = "logistic",
23    "formula" = atrisk ~ mathpost + condition + frlunch + efficacy +
24    mathpre, nodes = nodes.atrisk)
25  model.stanread <- list("model" = "linreg",
26    "formula" = stanread ~ atrisk + mathpost + condition + frlunch +
27    efficacy + mathpre, nodes = nodes.stanread)
```

The `mdmb` package views `stanread` (the auxiliary variable in the final regression model) as the ultimate "dependent" variable, and it considers all other variables "independent variables". Starting on line 28, the final part of the code combines the independent variable models into a list. On line 31, the data frame and the predictor list are passed into the `frm_em` function, which fits the sequence of models. Finally, the `summary` function on line 32 requests tables of parameter estimates.

R Script Ex3.1.R, continued

```
predictor.models <- list(frlunch = model.frlunch,
  efficacy = model.efficacy, mathpost = model.mathpost,
  atrisk = model.atrisk)
fit <- frm_em(dat = data, dep = model.stanread,
  ind = predictor.models)
summary(fit)
```

R Output

The `mdmb` output includes a table of results for every fitted regression model. The supporting model parameters are not of substantive interest, and they do not need to be reported. The output below shows the parameter estimates from the focal regression model. The first two columns display the unstandardized estimates and their standard errors, the third and fourth columns display the corresponding *t*-statistics and *p*-values, and the rightmost columns contain 95% confidence interval limits.

```
Model 3: stats::lm( mathpost ~ condition + frlunch + efficacy + mathpre )

  index       dv                    parm ON     est     se       t      p lower95 upper95
1    15 mathpost mathpost ON (Intercept)  1 29.0504 3.0085  9.6562 0.0000 23.1539 34.9469
2    16 mathpost   mathpost ON condition  1  2.2939 1.0226  2.2431 0.0249  0.2895  4.2982
3    17 mathpost     mathpost ON frlunch  1 -5.2352 1.0592 -4.9427 0.0000 -7.3111 -3.1593
4    18 mathpost    mathpost ON efficacy  1  0.7966 0.3391  2.3490 0.0188  0.1319  1.4612
5    19 mathpost     mathpost ON mathpre  1  0.5200 0.0607  8.5687 0.0000  0.4011  0.6390
6    20 mathpost          mathpost sigma  0  7.1076 0.3524 20.1691 0.0000  6.4169  7.7983

Explained variance R^2=0.4197
```

The results are interpreted in the same way as a complete-data regression analysis with categorical predictors. For example, the positive coefficient for the treatment assignment predictor indicates that, for two students who share the same covariate profile (i.e., lunch assistance, self-efficacy, and pretest scores), the model predicts that the student in the experimental condition should score 2.29 points higher than the student in the control group.

The corresponding test statistic indicates that the slope coefficient is statistically different from zero ($t = 2.24$, $p = .03$).

EXAMPLE 4: MODERATED REGRESSION WITH AN INTERACTION EFFECT

This example illustrates a multiple regression analysis with an incomplete interaction effect. The analysis uses the `behaviorachievement.dat` data set taken from a longitudinal study that followed 138 students from primary through middle school. The file includes three annual assessments of broad reading and math achievement beginning in the first grade, seventh grade standardized achievement test scores taken from a statewide assessment, and a final measure of broad reading and math obtained in ninth grade. The data also contain teacher ratings of behavioral symptoms and learning problems were also obtained in the first grade. The data description at the beginning of this document provides additional details. The variables for this analysis are as follows.

Name	Definition	Missing %	Scale
	Focal Variables		
BEHSYMP1	1st grade behavioral symptoms	3.6	Numeric
LRNPROB1	1st grade learning problems	2.2	Numeric
READ1	1st grade broad reading composite	6.5	Numeric
READ9	9th grade broad reading composite	17.4	Numeric
	Auxiliary Variables		
READ2	2nd grade broad reading composite	9.4	Numeric
STANREAD7	7th grade standardized math	19.6	Numeric

Analysis Model

The analysis model features ninth grade broad reading scores regressed on first grade reading achievement, teacher-rated learning problems and behavioral symptoms, and the product of first grade reading scores and learning problems.

$$READ_9 = \beta_0 + \beta_1(READ_1) + \beta_2(LRNPROB_1) + \beta_3(READ_1)(LRNPROB_1) + \beta_4(BEHSYMP_1^2) + \varepsilon \quad (8)$$

Moderated regression models (and models with non-linearities more generally) require a factored regression specification that splits the likelihood into separate parts for the outcome model and predictors.

Unlike a complete-data regression analysis, incomplete variables also require distributional assumptions and models that define those distributions. The analysis uses a factored regression specification that separates the likelihood function into separate components for each variable. In practical terms, this specification uses a sequence of univariate regression models to link incomplete predictors. The additional regression equations are as follows.

$$\begin{aligned} BEHSYMP_1 &= \gamma_{01} + \epsilon_1 \\ LRNPROB_1 &= \gamma_{02} + \gamma_{12}(BEHSYMP_1) + \epsilon_2 \\ READ_1 &= \gamma_{03} + \gamma_{13}(LRNPROB_1) + \gamma_{23}(BEHSYMP_1) + \epsilon_3 \end{aligned} \tag{9}$$

The composition of these models mimics the path diagram from Example 3.

The missing data literature often recommends an inclusive strategy that incorporates auxiliary variables that either predict missingness or correlate with the incomplete variables (Collins et al., 2001). Following earlier examples, auxiliary variables enter the model as additional outcomes that are predicted by the analysis variables and by each other. The additional regression equations are as follows.

$$\begin{aligned} READ_2 &= \gamma_{01} + \gamma_{11}(READ9GRP) + \gamma_{21}(READ_1) \\ &\quad + \gamma_{31}(LRNPROB_1) + \gamma_{41}(BEHSYMP_1) + \epsilon_1 \\ STANREAD_7 &= \gamma_{02} + \gamma_{12}(READ_2) + \gamma_{22}(READ9GRP) \\ &\quad + \gamma_{32}(READ_1) + \gamma_{42}(LRNPROB_1) + \gamma_{52}(BEHSYMP_1) + \epsilon_2 \end{aligned} \tag{10}$$

Along with the other models, the collection of regression equations can be viewed as a path model where the focal analysis is one part of a larger network. The key difference is that the path coefficients are just a tool for linking incomplete variables and do not represent a substantive theory.

R Script

Mplus and the `lavaan` package currently do not offer maximum likelihood estimation for incomplete interaction effects. The example instead uses the `mdmb` package. The R input file for the analysis is `Ex4.R`. The code block below shows the commands that import the data.

R Script Ex4.R

```
1   library(fdir)
2   library(lavaan)
3   library(mdmb)
4   set()
5   data <- read.table("behaviorachievement.dat", na.strings = "999")
6   names(data) <-c("id","male","hispanic","riskgrp","atrisk","behsymp1",
7     "lrnprob1","read1","read2","read3","read9","read9grp","stanread7",
8     "math1","math2","math3","math9","math9grp","stanmath7")
```

The example requires the `fdir`, `lavaan`, ad `mdmb` packages, which are loaded on lines 1 through 3. On line 4, the `set()` function of the `fdir` package identifies the file path of the folder containing the R script and sets this location as the working directory. On line 5, the `read.table` command imports the data, and the `na.strings` parameter specifies 999 as the global missing value code. It is only necessary to specify the name of the input data file. No file path is required when the data reside in the same folder as the R script as is the case here. Finally, variable names are listed beginning on line 6.

The analysis centers the two variables involved in the interaction at their grand means. Because the predictors are incomplete, the script uses `lavaan` to obtain maximum likelihood-estimated means for centering. The code block is shown below. The `model` variable on lines 9 and 10 defines a text string describing a set of empty regression models with only an intercept (the `~ 1` after each variable name). Along with the data frame, this model is passed into `lavaan`'s `inspectSampleCov` function on line 11. The resulting maximum likelihood estimates of the means are used to create new centered variables called `read1.cgm` and `lrnprob1.cgm` on lines 12 and 13.

R Script Ex4.1.R, continued

```
model <- "stanread7 ~ 1; read2 ~ 1; hispanic ~ 1; read9 ~ 1;
  read1 ~ 1; lrnprob1 ~ 1; behsymp1 ~ 1;"
descriptives <- inspectSampleCov(model, data, missing = "fiml")
data$read1.cgm <- data$read1 - descriptives$mean["read1"]
data$lrnprob1.cgm <- data$lrnprob1 - descriptives$mean["lrnprob1"]
summary(data[,c("stanread7","read2","read9","read1.cgm",
  "lrnprob1.cgm","behsymp1")])
```

The `mdmb` package requires the user to specify "nodes" or "pseudo-imputations" for the missing values. These nodes are essentially a fixed list of plausible score values that span each variable's range. Specifying these values is necessary for the optimization algorithm, which uses an imputation-like algorithm called numerical integration. The `summary` function on lines 14 and 15 generates a table displaying the observed values of the numeric variables. The summary table is as follows.

```
 stanread7         read2            read9          read1.cgm         lrnprob1.cgm          behsymp1
Min.   :100.0   Min.   : 20.00   Min.   : 41.00   Min.   :-47.1887   Min.   :-21.29959   Min.   :17.00
1st Qu.:228.0   1st Qu.: 83.00   1st Qu.: 81.00   1st Qu.:-12.1887   1st Qu.: -7.29959   1st Qu.:41.00
Median :263.0   Median : 92.00   Median : 89.00   Median : -0.1887   Median : -1.29959   Median :48.00
Mean   :264.5   Mean   : 93.74   Mean   : 88.55   Mean   :  0.6175   Mean   :  0.05597   Mean   :49.47
3rd Qu.:314.0   3rd Qu.:108.00   3rd Qu.: 97.00   3rd Qu.: 12.8113   3rd Qu.:  8.20041   3rd Qu.:58.00
Max.   :399.0   Max.   :150.00   Max.   :123.00   Max.   : 66.8113   Max.   : 35.70041   Max.   :92.00
NA's   :27      NA's   :13       NA's   :24       NA's   :9          NA's   :3           NA's   :5
```

The next part of the code creates variables that contain vectors of plausible replacement scores that span the entire range of the distributions. For continuous variables, specifying 20 to 40 nodes is usually sufficient. For example, `nodes.read1` is a vector of plausible centered scores ranging from –55 to 75 in increments of two, and `nodes.lrnprb1` is a sequence of centered scores between –30 and 50 in increments of two. To account for the possibility that the missing scores fall outside the observed range, the vectors specify values beyond the minimum and maximum scores from the data.

R Script Ex4.1.R, continued

```
10  nodes.stanread7 <- seq(80, 420, by = 5)
11  nodes.read2 <- seq(10, 160, by = 5)
12  nodes.read9 <- seq(30, 130, by = 2)
13  nodes.read1 <- seq(-55, 75, by = 2)
14  nodes.lrnprob1 <- seq(-30, 50, by = 2)
15  nodes.behsymp1 <- seq(10, 100, by = 2)
```

The next part of the script specifies a model for each analysis variable and auxiliary variable. The predictor variable regressions from Equation 9 are listed first, followed by the focal moderated regression model from Equation 8. The auxiliary variable regressions from Equation 10 are last. Each model object includes three arguments: the type of regression (linear or logistic), an equation, and the incomplete variable's vector of nodes or pseudo-imputations. Note that the focal model list beginning on line 25 includes the product of two centered variables.

R Script Ex4.1.R, continued

```
16  model.behsymp1 <- list( "model" = "linreg",
17    "formula" = behsymp1 ~ 1,
18    nodes = nodes.behsymp1)
19  model.lrnprob1 <- list( "model" = "linreg",
20    "formula" = lrnprob1.cgm ~ behsymp1,
21    nodes = nodes.lrnprob1)
22  model.read1 <- list( "model" = "linreg",
23    "formula" = read1.cgm ~ lrnprob1.cgm + behsymp1,
24    nodes = nodes.read1)
25  model.read9 <- list( "model" = "linreg",
26    "formula" = read9 ~ read1.cgm + lrnprob1.cgm +
27    read1.cgm*lrnprob1.cgm + behsymp1,
29    nodes = nodes.read9)
```

```
30  model.read2 <- list("model" = "linreg",
31    "formula" = read2 ~ read9 + read1.cgm + lrnprob1.cgm + behsymp1,
32    nodes = nodes.read2)
33  model.stanread7 <- list("model" = "linreg",
34    "formula" = stanread7 ~ read2 + read9 + read1.cgm
35    + lrnprob1.cgm + behsymp1, nodes = nodes.stanread7)
```

The `mdmb` package views `stanread7` (the auxiliary variable in the final regression model) as the ultimate "dependent" variable in the sequence, and it considers all other variables as "independent variables". Starting on line 36, the final part of the code combines the independent variable models into a list. On line 39, the data frame and the predictor list are passed into the `frm_em` function, which fits the sequence of models. Finally, the `summary` function on line 41 requests tables of parameter estimates.

R Script Ex4.1.R, continued

```
36  predictor.models <- list(behsymp1 = model.behsymp1, lrnprob1 =
37    model.lrnprob1, read1 = model.read1, read9 = model.read9,
38    read2 = model.read2)
39  fit <- frm_em(dat = data, dep = model.stanread7, ind =
40    predictor.models)
41  summary(fit)
```

R Output

The `mdmb` output includes a table of results for every fitted regression model. The supporting model parameters are not of substantive interest, and they do not need to be reported. The output below shows the parameter estimates from the focal model. The first two columns display the unstandardized estimates and their standard errors, the third and fourth columns display the corresponding *t*-statistics and *p*-values, and the rightmost columns contain 95% confidence interval limits.

```
Model 4: stats::lm( read9 ~ read1.cgm + lrnprob1.cgm + read1.cgm * lrnprob1.cgm + behsymp1 )

  index    dv                              parm ON     est     se       t      p lower95  upper95
1    14 read9            read9 ON (Intercept)  1 94.5840 4.9630 19.0577 0.0000 84.8566 104.3114
2    15 read9              read9 ON read1.cgm  1  0.5182 0.0413 12.5350 0.0000  0.4372   0.5992
3    16 read9           read9 ON lrnprob1.cgm  1 -0.2913 0.1144 -2.5455 0.0109 -0.5155  -0.0670
4    17 read9               read9 ON behsymp1  1 -0.1396 0.0990 -1.4103 0.1585 -0.3335   0.0544
5    18 read9 read9 ON read1.cgm:lrnprob1.cgm  1  0.0126 0.0044  2.8549 0.0043  0.0040   0.0213
6    19 read9                     read9 sigma  0  8.9828 0.6140 14.6292 0.0000  7.7793  10.1863

Explained variance R^2=0.6353
```

The lower-order terms in a moderated regression are conditional effects that depend on scaling or centering. Specifically, the lower-order slope of first grade reading scores ($\hat{\beta}_1 = 0.52$) is the effect of that predictor at the mean of the first-grade learning problems, and the learning problems slope ($\hat{\beta}_2 = -0.29$) similarly reflects a conditional effect at the reading score mean. The interaction slope captures the change in the first-grade reading slope for each one-unit increase in learning problems (and vice versa). Specifically, the positive coefficient ($\hat{\beta}_3 = 0.013$) indicates that the association between first and ninth grade reading scores becomes stronger (i.e., more positive) as learning problems increase. That is, the predictive power of early reading on later reading is strongest for students with elevated learning problem ratings in first grade.

EXAMPLE 5: CURVILINEAR REGRESSION

This example illustrates a multiple regression analysis with an incomplete curvilinear effect. The analysis uses the `mathachievement.dat` data set taken from an educational intervention where 250 students were assigned to an intervention and comparison condition. The file includes pretest and posttest math achievement scores, a measure of math self-efficacy, standardized reading scores taken from a statewide assessment, and several sociodemographic variables. The analysis variables are as follows.

Name	Definition	Missing %	Scale
	Focal Variables		
MATHPOST	Math achievement posttest	18.0	Numeric
ANXIETY	Math anxiety composite	8.4	Numeric
FRLUNCH	Lunch assistance code	4.4	0 = None, 1 = Free/reduced lunch
EFFICACY	Math self-efficacy rating	9.6	Ordinal (1 to 6)
MATHPRE	Math achievement pretest	0	Numeric
	Auxiliary Variables		
ATRISK	Behavioral disorder risk	5.2	0 = Low risk, 1 = At-risk
STANREAD	Standardized reading	9.2	Numeric

Analysis Model

The analysis model features math posttest scores regressed on anxiety and its square, the lunch assistance dummy code, math self-efficacy ratings, and math pretest scores.

$$\begin{aligned} MATHPOST &= \beta_0 + \beta_1(ANXIETY) + \beta_2(ANXIETY^2) \\ &+ \beta_3(FRLUNCH) + \beta_4(EFFICACY) + \beta_5(MATHPRE) + \varepsilon \end{aligned} \tag{11}$$

Curvilinear regression models (and models with non-linearities more generally) require a factored regression specification that splits the likelihood into separate parts for the outcome model and predictors.

Unlike a complete-data regression analysis, incomplete variables also require distributional assumptions and models that define those distributions. The analysis uses a factored regression specification that separates the likelihood function into separate components for each variable. In

practical terms, this specification uses a sequence of univariate regression models to link incomplete predictors. The additional regression equations, two of which are logistic models, are as follows.

$$\begin{aligned} \text{logit}(FRLUNCH) &= \gamma_{01} + \gamma_{11}(MATHPRE) \\ \text{logit}(EFFICACY) &= \gamma_{02} + \gamma_{12}(FRLUNCH) + \gamma_{22}(MATHPRE) \\ ANXIETY &= \gamma_{03} + \gamma_{13}(EFFICACY) + \gamma_{23}(FRLUNCH) + \gamma_{33}(MATHPRE) + \epsilon_3 \end{aligned} \tag{12}$$

These equations essentially comprise a path model where math pretest scores predict the lunch assistance indicator, the lunch assistant dummy code and math pretest scores predict efficacy, and all three variables, in turn, predict anxiety (see the path diagram in Example 3). The complete variable is always on the right side of regression equations because it does not require a model.

The missing data literature often recommends an inclusive strategy that incorporates auxiliary variables that either predict missingness or correlate with the incomplete variables (Collins et al., 2001). Following earlier examples, auxiliary variables enter the model as additional outcomes that are predicted by the analysis variables and by each other. The additional regression equations are as follows.

$$\begin{aligned} \text{logit}(ATRISK) &= \gamma_{04} + \gamma_{14}(MATHPOST) + \gamma_{24}(ANXIETY) \\ &\quad + \gamma_{34}(FRLUNCH) + \gamma_{44}(EFFICACY) + \gamma_{54}(MATHPRE) \\ STANREAD &= \gamma_{05} + \gamma_{15}(ATRISK) + \gamma_{25}(MATHPOST) + \gamma_{35}(ANXIETY) \\ &\quad + \gamma_{45}(FRLUNCH) + \gamma_{55}(EFFICACY) + \gamma_{65}(MATHPRE) + \epsilon_5 \end{aligned} \tag{13}$$

Again, the entire collection of regression equations can be viewed as a path model where the curvilinear regression is one piece of a larger network. The key difference is that the path coefficients are just a tool for linking incomplete variables and do not represent a substantive theory.

R Script

Mplus and the `lavaan` package currently do not offer maximum likelihood estimation for incomplete interaction effects. The example instead uses the `mdmb` package. The R input file for the analysis is `Ex5.R`. The code block below shows the commands that import the data.

R Script Ex5.R

```
1   library(fdir)
2   library(lavaan)
3   library(mdmb)
4   set()
5   data <- read.table("behaviorachievement.dat", na.strings = "999")
6   names(data) <-c("id","male","hispanic","riskgrp","atrisk","behsymp1",
7     "lrnprob1","read1","read2","read3","read9","read9grp","stanread7",
8     "math1","math2","math3","math9","math9grp","stanmath7")
```

The example requires the `fdir`, `lavaan`, ad `mdmb` packages, which are loaded on lines 1 through 3. On line 4, the `set()` function of the `fdir` package identifies the file path of the folder containing the R script and sets this location as the working directory. On line 5, the `read.table` command imports the data, and the `na.strings` parameter specifies 999 as the global missing value code. It is only necessary to specify the name of the input data file. No file path is required when the data reside in the same folder as the R script as is the case here. Finally, variable names are listed beginning on line 6.

The analysis centers math anxiety predictor (the variable involved in the quadratic effect) at its grand mean. Because the predictors are incomplete, the script uses `lavaan` to obtain maximum likelihood-estimated means for centering. The `model` variable on lines 9 and 10 of the following code block defines a text string describing a set of empty regression models with only an intercept (the `~ 1` after each variable name). Along with the data frame, this model is passed into `lavaan`'s `inspectSampleCov` function on line 11. The resulting maximum likelihood estimate of the mean is used to create a new centered variable called `anxiety.cgm`.

R Script Ex5.1.R, continued

```
 9  model <- "stanread ~ 1; atrisk ~ 1; mathpost ~ 1; anxiety ~ 1;
10    frlunch ~ 1; efficacy ~ 1; mathpre ~ 1;"
11  descriptives <- inspectSampleCov(model, data, missing = "fiml")
12  data$anxiety.cgm <- data$anxiety - descriptives$mean["anxiety"]
13  summary(data[,c("stanread","mathpost","anxiety.cgm","efficacy")])
```

The `mdmb` package requires the user to specify "nodes" or "pseudo-imputations" for the missing values. These nodes are essentially a list of plausible score values or pseudo-imputations that span each variable's range. Specifying these values is necessary for the optimization algorithm, which uses an imputation-like algorithm called numerical integration. The `summary` function on line 13 generates a table displaying the observed values from the data. The summary table is as follows.

```
    stanread        mathpost      anxiety.cgm          efficacy
 Min.   :27.00   Min.   :37.00   Min.   :-18.2628   Min.   :1.000
 1st Qu.:45.00   1st Qu.:52.00   1st Qu.: -5.2628   1st Qu.:2.000
 Median :55.00   Median :57.00   Median : -1.2628   Median :3.000
 Mean   :52.52   Mean   :57.45   Mean   : -0.1056   Mean   :3.394
 3rd Qu.:60.50   3rd Qu.:63.00   3rd Qu.:  3.7372   3rd Qu.:5.000
 Max.   :69.00   Max.   :85.00   Max.   : 25.7372   Max.   :6.000
 NA's   :23      NA's   :45      NA's   :21         NA's   :24
```

The next part of the code creates variables that contain vectors of plausible replacement scores (nodes, pseudo-imputations) that span the entire range of the distributions. For continuous variables, specifying 20 to 40 nodes is usually sufficient. For example, `nodes.mathpost` is a sequence of raw scores between 30 and 90 in increments of two, and `nodes.anxiety` is a vector of plausible centered scores ranging from –30 to 30 in increments of two. To account for the possibility that the missing scores fall outside the observed range, the vectors specify values beyond the minimum and maximum scores from the data.

R Script Ex5.1.R, continued

```
nodes.frlunch <- c(0,1)
nodes.efficacy <- seq(1, 6, by = 1)
nodes.mathpost <- seq(30, 90, by = 2)
nodes.anxiety <- seq(-30, 30, by = 2)
nodes.atrisk <- c(0,1)
nodes.stanread <- c(20, 80, by = 2)
```

The next part of the script specifies a model for each analysis variable and auxiliary variable. The predictor variable regressions from Equation 12 are listed first, followed by the focal model from Equation 11. The auxiliary variable regressions from Equation 13 are last. Each model object includes three arguments: the type of regression (linear or logistic), an equation, and the incomplete variable's vector of nodes or pseudo-imputations. Linear regressions are specified with `"model" = "linreg"` parameter, and the binary logistic regression is specified using `"model" = "logistic"`. Note that the focal model list beginning on line 29 includes the square of the centered variable (i.e., `I(anxiety.cgm^2)`).

R Script Ex5.1.R, continued

```
model.frlunch <- list("model" = "logistic",
  "formula" = frlunch ~ mathpre,
  nodes = nodes.frlunch)
model.efficacy <- list("model" = "linreg",
  "formula" = efficacy ~ frlunch + mathpre,
  nodes = nodes.efficacy)
model.anxiety <- list("model" = "linreg",
  "formula" = anxiety ~ efficacy + frlunch + mathpre,
  nodes = nodes.anxiety)
model.mathpost <- list("model" = "linreg",
  "formula" = mathpost ~ anxiety.cgm + I(anxiety.cgm^2) +
  frlunch + efficacy + mathpre, nodes = nodes.mathpost)
```

```
32  model.atrisk <- list("model" = "logistic",
33    "formula" = atrisk ~ mathpost + anxiety + frlunch + efficacy +
34    mathpre, nodes = nodes.atrisk)
35  model.stanread <- list("model" = "linreg",
36    "formula" = stanread ~ atrisk + mathpost + anxiety + frlunch +
37    efficacy + mathpre, nodes = nodes.stanread)
```

The `mdmb` package views `stanread` (the auxiliary variable in the final regression model) as the ultimate "dependent" variable in the sequence, and it considers all other variables "independent variables". Starting on line 38, the final part of the code combines the independent variable models into a list. On line 41, the data frame and the predictor list are passed into the `frm_em` function, which fits the sequence of models. Finally, the `summary` function on line 43 requests tables of parameter estimates.

R Script Ex5.1.R, continued

```
38  predictor.models <- list(frlunch = model.frlunch,
39    efficacy = model.efficacy, anxiety = model.anxiety,
40    mathpost = model.mathpost, atrisk = model.atrisk)
41  fit <- frm_em(dat = data, dep = model.stanread,
42    ind = predictor.models)
43  summary(fit)
```

R Output

The `mdmb` output includes a table of results for every fitted regression model. The supporting model parameters are not of substantive interest, and they do not need to be reported. The output below shows the parameter estimates from the focal curvilinear model. The first two columns display the unstandardized estimates and their standard errors, the third and fourth columns display the corresponding *t*-statistics and *p*-values, and the rightmost columns contain 95% confidence interval limits.

```
Model 4: stats::lm( mathpost ~ anxiety.cgm + I(anxiety.cgm^2) + efficacy + frlunch + mathpre )

  index       dv                          parm ON     est     se       t      p lower95 upper95
1    15 mathpost      mathpost ON (Intercept)  1 33.2388 3.3678  9.8695 0.0000 26.6380 39.8396
2    16 mathpost      mathpost ON anxiety.cgm  1  0.0398 0.0793  0.5015 0.6160 -0.1156  0.1952
3    17 mathpost mathpost ON I(anxiety.cgm^2)  1 -0.0209 0.0059 -3.5452 0.0004 -0.0324 -0.0093
4    18 mathpost         mathpost ON efficacy  1  1.0629 0.3324  3.1975 0.0014  0.4114  1.7145
5    19 mathpost          mathpost ON frlunch  1 -5.5373 1.0398 -5.3255 0.0000 -7.5752 -3.4994
6    20 mathpost          mathpost ON mathpre  1  0.4648 0.0651  7.1361 0.0000  0.3371  0.5925
7    21 mathpost               mathpost sigma  0  6.9386 0.3460 20.0511 0.0000  6.2604  7.6168
```

In a curvilinear regression model, the lower-order term for math anxiety is a conditional effect that depends on scaling or centering. The slope conveys the instantaneous linear change in the outcome at the anxiety mean, controlling for all other predictors ($\hat{\beta}_1 = 0.04$). The negative quadratic coefficient ($\hat{\beta}_2 = -0.02$) indicates that the positive association at the mean decreases (i.e., becomes less positive) as anxiety increases (and vice versa). At high enough levels of anxiety, the association becomes negative, such that anxiety has a debilitating effect on math performance.

SECTION 2: BAYESIAN ESTIMATION AND MODEL-BASED MULTIPLE IMPUTATION

EXAMPLE 6: MULTIPLE REGRESSION WITH MULTIVARIATE NORMAL DATA

This example illustrates a multiple regression analysis with multivariate normal incomplete data. The analysis uses the `behaviorachievement.dat` data set taken from a longitudinal study that followed 138 students from primary through middle school. The file includes three annual assessments of broad reading and math achievement beginning in the first grade, seventh grade standardized achievement test scores taken from a statewide assessment, and a final measure of broad reading and math obtained in ninth grade. The data also contain teacher ratings of behavioral symptoms and learning problems were also obtained in the first grade. The data description at the beginning of this document provides additional details. The variables for this analysis are as follows.

Name	Definition	Missing %	Scale
	Focal Variables		
BEHSYMP1	1st grade behavioral symptoms	3.6	Numeric
LRNPROB1	1st grade learning problems	2.2	Numeric
READ1	1st grade broad reading composite	6.5	Numeric
READ9	9th grade broad reading composite	17.4	Numeric
	Auxiliary Variables		
READ2	2nd grade broad reading composite	9.4	Numeric
STANREAD7	7th grade standardized math	19.6	Numeric

Analysis Model

The analysis model features ninth grade broad reading scores regressed on first grade reading achievement and teacher-rated learning problems and behavioral symptoms.

$$READ_9 = \beta_0 + \beta_1(READ_1) + \beta_2(LRNPROB_1) + \beta_3(BEHSYMP_1^{\prime}) + \varepsilon \qquad (14)$$

Unlike a complete-data regression analysis, all incomplete variables require distributional assumptions, including the predictors. By default, Blimp invokes a multivariate normal distribution for predictors.

The missing data literature often recommends an inclusive strategy that incorporates auxiliary variables that either predict missingness or correlate with the incomplete variables (Collins et al., 2001). Following the same factored regression specification from earlier examples (e.g., Examples 2 through 6), auxiliary variables enter the model as additional outcomes that are predicted by the analysis variables and by each other. The additional regression equations are as follows.

$$\begin{aligned} READ_2 &= \gamma_{01} + \gamma_{11}(READ9GRP) + \gamma_{21}(READ_1) \\ &+ \gamma_{31}(LRNPROB_1) + \gamma_{41}(BEHSYMP_1) + \epsilon_1 \\ STANREAD_7 &= \gamma_{02} + \gamma_{12}(READ_2) + \gamma_{22}(READ9GRP) \\ &+ \gamma_{32}(READ_1) + \gamma_{42}(LRNPROB_1) + \gamma_{52}(BEHSYMP_1) + \epsilon_2 \end{aligned} \tag{15}$$

Along with the focal regression model from Equation 14, the collection of regressions can be viewed as a path model, where the focal regression is one part of a larger network (see the path diagram from Example 2). The key difference is that the path coefficients are just a tool for linking incomplete variables and do not represent a substantive theory.

Blimp Script

The code block below shows Blimp script `Ex6.1.inp`.

Blimp Script Ex6.1.imp

```
DATA: behaviorachievement.dat;
VARIABLES: id male hispanic riskgrp atrisk behsymp1 lrnprob1
  read1 read2 read3 read9 read9grp stanread7
  math1 math2 math3 math9 math9grp stanmath7;
MISSING: 999;
MODEL:
read9 ~ read1@beta1 lrnprob1@beta2 behsymp1@beta3;
stanread7 read2 ~ read9 read1 lrnprob1 behsymp1;
```

```
 9  TEST:
10  beta1:beta3 = 0;
11  SEED: 90291;
12  BURN: 1000;
13  ITERATIONS: 10000;
```

The first five lines can be viewed as a set of commands that specify information about the data and variables. The `DATA` command specifies the name of the input text file. No file path is required when the data file is located in the same directory as the script, as it is here. Starting on line 2, the `VARIABLES` command names the data columns, and the `MISSING` command on line 5 defines a global missing value code as 999.

The `MODEL` and `TEST` blocks can be viewed as a set. The `MODEL` command lists the regression models, with outcome variables to the left of the tilde and predictors to the right. Line 7 assigns labels the slope coefficients using the @ symbol. Blimp automatically configures the explanatory variable models under the assumption that they are normally distributed. Line 8 is a syntax shortcut that produces the two auxiliary variable regression models in Equation 15; in the first model, `READ2` is regressed on the focal variables, and the second model features `STANREAD7` regressed on `READ2` and the focal variables. The `TEST` command uses the parameter labels to specify a custom hypothesis test that all three slopes equal zero. This command produces the Bayesian Wald test (Asparouhov & Muthén, 2021), which is essentially a chi-square statistic that captures the discrepancy between the Bayesian point estimates (posterior means) and the hypothesized values of zero.

Lines 11 through 13 can be viewed as a block of commands that specify features of the MCMC algorithm: the `SEED` command gives an integer string that initializes the random number generator, the `BURN` command specifies the number of iterations for the warm-up or burn-in period, and the `ITERATIONS` command gives the number of MCMC iterations on which the analysis summaries are based (essentially, the number of MCMC cycles following the warm-up period).

Blimp prints a table of regression results for each outcome variable to the left of a tilde, and it orders the tables alphabetically. In this example, the focal model's table would not appear first on the output. Blimp allows users to order tables by assigning labels to blocks of regression equations. To illustrate, the code block below assigns the label `focal.model` to main regression and the label `auxiliary.models` to the auxiliary variable regressions. Because output tables are

listed in the same order as the labels, the focal results would now appear before the ancillary model results.

```
MODEL:
focal.model:
read9 ~ read1@beta1 lrnprob1@beta2 behsymp1@beta3;
auxiliary.models:
stanread7 read2 ~ read9 read1 lrnprob1 behsymp1;
```

Blimp Output

Prior to inspecting the parameter estimates, it is important to investigate the potential scale reduction (PSR) factor diagnostics (Gelman & Rubin, 1992) to determine whether MCMC has converged. Blimp divides the burn-in period into 20 equal segments, and it computes the PSR diagnostic for every parameter. The table located near the top of the output reports the highest (worst) PSR value across all parameters in every model. A common recommendation is that these values should be less than 1.05 or perhaps 1.10 (Asparouhov & Muthén, 2010a; Gelman et al., 2014). If the PSR in the bottom row of the table (the final check of the burn-in period) is above these cutoffs, then rerun the analysis with a longer burn-in period.

```
BURN-IN POTENTIAL SCALE REDUCTION (PSR) OUTPUT:

  NOTE: Split chain PSR is being used. This splits each chain's
        iterations to create twice as many chains.

  Comparing iterations across 2 chains     Highest PSR   Parameter #
                            26 to 50             1.263            15
                            51 to 100            1.081            41
                            76 to 150            1.056            37
                           101 to 200            1.037            26
                           126 to 250            1.059            32
                           151 to 300            1.027            17
                           176 to 350            1.031            41
                           201 to 400            1.022            33
                           226 to 450            1.034            17
```

```
                    251 to 500          1.020          15
                    276 to 550          1.027          20
                    301 to 600          1.023          44
                    326 to 650          1.014          19
                    351 to 700          1.010          45
                    376 to 750          1.014          33
                    401 to 800          1.012          33
                    426 to 850          1.017          37
                    451 to 900          1.023          41
                    476 to 950          1.025          41
                    501 to 1000         1.016          41
```

The next output excerpt shows information about the variables in the analysis and the models used for estimation. The `MODELS` summary section is reserved for outcome variables that appear to the left of a tilde symbol. In this example, Blimp automatically constructs supporting models for incomplete predictor variables, so these models are omitted from the table.

```
DATA INFORMATION:

  Sample Size:              138
  Missing Data Rates:

                   read9 = 17.39
                   read2 = 09.42
               stanread7 = 19.57
                behsymp1 = 03.62
                lrnprob1 = 02.17
                   read1 = 06.52
```

```
MODEL INFORMATION:

  NUMBER OF PARAMETERS
    Outcome Models:         18
    Predictor Models:       12

  PREDICTORS
    Incomplete continuous:  behsymp1 lrnprob1 read1

MODELS

   focal.model:
    [1]  read9 ~ Intercept read1@beta1 lrnprob1@beta2 behsymp1@beta3

   auxiliary.models:
    [2]  read2 ~ Intercept read9 read1 lrnprob1 behsymp1
    [3]  stanread7 ~ Intercept read2 read9 read1 lrnprob1 behsymp1
```

Most software programs that fit regression models report an omnibus *F* test that evaluates the set of slope coefficients. The `TEST` command in the previous script requested an analogous Bayesian Wald chi-square statistic (Asparouhov & Muthén, 2021) that evaluates the null hypothesis that all population slopes equal zero. The chi-square statistic, degrees of freedom, and *p*-value appear near the bottom of the `MODEL FIT` section under the `WALD TEST` heading. The test statistic is statistically significant, thus refuting the null hypothesis.

```
MODEL FIT:

  INFORMATION CRITERIA

    Marginal Likelihood
      DIC2                  3424.672
      WAIC                  3458.337
```

```
Conditional Likelihood
      DIC2                  3424.672
      WAIC                  3458.337

WALD TESTS (Asparouhov & Muthén, 2021)

  Test #1

    Full:
      [1]  read9 ~ Intercept read1@beta1 lrnprob1@beta2 behsymp1@beta3

    Restricted:
      [1]  read9 ~ Intercept read1@beta1 lrnprob1@beta2 behsymp1@beta3

    Constraints in Restricted:
      [1]  beta1 = 0
      [2]  beta2 = 0
      [3]  beta3 = 0


    Wald Statistic (Chi-Square)               166.865
    Number of Parameters Tested (df)                3
    Probability                                 0.000
```

The tables summarizing the focal regression model includes unstandardized coefficients, standardized slopes, and variance explained effect size estimates. MCMC estimation produces a distribution for each parameter in the table. The median and standard deviation columns describe the center and spread of the posterior distributions; although they make no reference to drawing repeated samples, they are analogous—and numerically equivalent in most cases—to frequentist point estimates and standard errors. The 95% credible intervals in the rightmost columns give a range that captures 95% of the parameter's distribution. These are akin to confidence intervals, but the intervals describe parameter distributions rather than characteristics of repeated samples. The `N_Eff` values in rightmost column of the table give the effective number of MCMC samples for each parameter. These quantities essentially represent the number of independent estimates on which the parameter summaries are based after removing autocorrelations from the MCMC process. Gelman et al. (2014, p. 287) recommend values greater than 100. All values in the example table exceed this recommended minimum. In cases

where the N_Eff values are insufficient, increasing the value on the ITERATIONS command will remedy the issue. The table summarizing the focal regression model is shown below.

```
OUTCOME MODEL ESTIMATES:

  Summaries based on 10000 iterations using 2 chains.

  focal.model block:

Outcome Variable:  read9

Parameters                          Median     StdDev       2.5%      97.5%        PSR      N_Eff
                                ------------------------------------------------------------------
Variances:
  Residual Var.                     91.260     12.801     70.740    120.580      1.001   6116.298

Coefficients:
  Intercept                         66.006      6.051     53.949     77.819      1.000   5311.995
  read1                              0.504      0.044      0.421      0.591      1.000   6268.863
  lrnprob1                          -0.247      0.120     -0.479     -0.009      1.000   5248.396
  behsymp1                          -0.182      0.105     -0.387      0.025      1.001   6089.990

Standardized Coefficients:
  read1                              0.689      0.040      0.603      0.757      1.000   5630.823
  lrnprob1                          -0.177      0.085     -0.342     -0.006      1.000   5208.179
  behsymp1                          -0.147      0.083     -0.305      0.020      1.001   6088.430

Proportion Variance Explained
  by Coefficients                    0.595      0.050      0.487      0.680      1.001   5806.322
  by Residual Variation              0.405      0.050      0.320      0.513      1.001   5806.322

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

The results are interpreted in the same way as a complete-data regression analysis. For example, consider the first-grade reading score slope. The model predicts that two individuals who differ by one point on READ1 but are the same on LRNPROB1 and BEHSYMP1 should differ by 0.50 points on READ9. The 95% credible interval limits suggest this effect is statistically different from zero ($p < .05$) because the null value is well outside the interval. The standardized coefficients convey the expected change in standard deviation units for a one standard deviation increase in a given predictor. For example, the model predicts that two individuals who differ by one standard deviation on READ1 but are the same on LRNPROB1 and BEHSYMP1 should differ by .69 standard deviations on READ9. Collectively, the predictors explain 60% of the variation in ninth-grade reading scores. Note that the tabled values are numerically identical to the maximum likelihood estimates from Example 1.

The Blimp output also includes tables of regression model parameters for the auxiliary variables as well as the auto-generated models for incomplete predictors. These additionally results are not of substantive interest and would not be reported. The auxiliary variable models appear in `OUTCOME MODEL ESTIMATES` section with the focal results, and the auto-generated predictor models are displayed under the heading `PREDICTOR MODEL ESTIMATES`.

Saving Multiple Imputations

MCMC estimation imputes missing values at every iteration, such that the resulting Bayesian estimates average over thousands of plausible replacement scores (10,000 sets in this example). A subset of the imputations can be saved for reanalysis in the frequentist framework, if desired. The Blimp input file `Ex6.2.imp` is identical `Ex6.1.imp`, but it adds the following lines at the bottom of the script.

```
NIMPS: 20;
CHAINS: 20;
SAVE:
stacked = ./imps/imps.dat;
separate = ./imps/imp*.dat;
```

The `NIMPS`, `CHAINS` and `SAVE` commands can be viewed as a set. Setting `NIMPS` equal to `CHAINS` saves a single filled-in data set from the final iteration of a unique `MCMC` process, thus avoiding autocorrelation among the imputations. The `SAVE` command provides a name for the imputed data sets. The script illustrates how to save data sets in two common formats. The stacked keyword creates a stacked file where all imputations are in a single file, and the separate keyword saves each imputed data set to a separate file with the asterisk replaced by a numeric index. To keep things organized, the `./imps` part of the file path points to a subfolder named `imps` located within the same folder as the script and data. The separate keyword also creates a list of file names needed for analysis in Mplus (in this example, a filed called `implist.dat` located in the `imps` folder).

When saving imputations, the bottom of the Blimp output file displays a table listing the order of the variables in the output data sets. All variables are saved regardless of whether they appeared in the fitted models. When saving data to a stacked file (e.g., for analysis in R or other packages), the first variable in the file is an integer index that identifies which data set each row belongs to (e.g., an integer variable that ranges from 1 to 20 in this example).

```
VARIABLE ORDER IN IMPUTED DATA:

separate = './imps/imp*.dat'

   id male hispanic riskgrp atrisk behsymp1 lrnprob1 read1 read2 read3
  read9 read9grp stanread7 math1 math2 math3 math9 math9grp stanmath7

stacked = './imps/imps.dat'

   imp# id male hispanic riskgrp atrisk behsymp1 lrnprob1 read1 read2 read3
   read9 read9grp stanread7 math1 math2 math3 math9 math9grp stanmath7
```

The imputed data sets are subsequently analyzed in another software package, and estimates and standard errors are combined using Rubin's rules (Little & Rubin, 2020). The analysis phase does not utilize the auxiliary variables, as their information is embedded in the imputations. Scripts for analyzing the imputed data sets are found in the next subsections.

Analyzing Imputations in Mplus

In lieu of the Bayesian estimates, Blimp's `SAVE` command can be used to save multiple imputations for analysis in the frequentist framework. Returning to the previous Blimp script, the `SAVE` command and the `separate` keyword saved each imputed data set to a separate file with the asterisk replaced by a numeric index. The separate keyword also creates a list of file names needed for analysis in Mplus (in this example, a filed called `implist.dat` located in the `imps` subfolder). The contents of this file are as follows.

```
imp1.dat
imp2.dat
imp3.dat
imp4.dat
imp5.dat
imp6.dat
imp7.dat
imp8.dat
imp9.dat
```

```
imp10.dat
imp11.dat
imp12.dat
imp13.dat
imp14.dat
imp15.dat
imp16.dat
imp17.dat
imp18.dat
imp19.dat
imp20.dat
```

The Mplus input file for analyzing the imputations is `Ex6.inp`. The script is virtually identical to the `Ex6.1.inp` file described in Example 1 with three exceptions. First, instead of naming the raw data set, the `DATA` command lists the text file containing the names of the imputed data sets (the `implist.dat` file located in the `./imps` subdirectory). The `type = imputation` subcommand instructs Mplus that the input data is a list of file names. Second, the missing subcommand is omitted because the analysis variables are now complete. Finally, the `MODEL` section no longer specifies a normal distribution for the predictors. Readers can refer back to Example 1 for a detailed description of the other commands. The code block below shows the analysis and pooling script.

Mplus Script Ex6.inp

```
DATA:
file = ./imps/implist.dat;
type = imputation;
VARIABLE:
names = id male hispanic riskgrp atrisk behsymp1 lrnprob1
   read1 read2 read3 read9 read9grp stanread7
   math1 math2 math3 math9 math9grp stanmath7;
usevariables = read9 read1 lrnprob1 behsymp1;
MODEL:
read9 on read1 lrnprob1 behsymp1 (beta1-beta3);
```

```
MODEL TEST:
0 = beta1; 0 = beta2; 0 = beta3;
OUTPUT:
stdyx cinterval;
```

Mplus Output

When fitting regression models to complete data sets, researchers often use an omnibus *F* test to evaluate the set of slope coefficients. The `MODEL TEST` command specified a multiple imputation Wald chi-square statistic evaluating the null hypothesis that the population slopes equal 0 (Asparouhov & Muthén, 2010b). The chi-square statistic, degrees of freedom, and *p*-value appear near the bottom of the `MODEL FIT INFORMATION` section under the `Wald Test of Parameter Constraints` heading. The test statistic is statistically significant, thus refuting the null hypothesis.

```
MODEL FIT INFORMATION

Number of Free Parameters                       5

...

Wald Test of Parameter Constraints

          Value                            175.893
          Degrees of Freedom                     3
          P-Value                           0.0000
```

The table of unstandardized parameter estimates is shown below. The first two columns display the pooled unstandardized estimates and standard errors, and the third and fourth columns display the corresponding *z*-statistics and *p*-values. The focal model results are shown in bold typeface. The `Rate of Missing` column (also called the fraction of missing information in the literature) quantifies the imputation noise in each estimate as proportion of its squared standard error.

```
MODEL RESULTS

                                                    Two-Tailed   Rate of
                  Estimate       S.E.  Est./S.E.    P-Value    Missing

 READ9    ON
    READ1            0.506      0.043     11.868      0.000      0.182
    LRNPROB1        -0.231      0.113     -2.047      0.041      0.149
    BEHSYMP1        -0.189      0.101     -1.864      0.062      0.160

 Intercepts
    READ9           65.487      5.803     11.284      0.000      0.150

 Residual Variances
    READ9           86.366     11.202      7.710      0.000      0.138
```

The results are interpreted in the same way as a complete-data regression analysis. For example, consider the first-grade reading score slope. The model predicts that two individuals who differ by one point on `READ1` but are the same on `LRNPROB1` and `BEHSYMP1` should differ by .51 points on `READ9`. The corresponding test statistic indicates that the slope coefficient is statistically different from zero ($z = 11.87$, $p < .001$). Note that these estimates are numerically identical to those from Bayesian and maximum likelihood estimation.

Specifying the `stdyx` keyword with the `OPTIONS` command prints the table of standardized estimates and R-squared statistics shown below. The slope coefficients convey the expected change in standard deviation units for a one standard deviation increase in a given predictor. For example, the model predicts that two individuals who differ by one standard deviation on `READ1` but are the same on `LRNPROB1` and `BEHSYMP1` should differ by .70 standard deviations on `READ9`. Collectively, the predictors explain 61% of the variation in ninth-grade reading scores.

```
STANDARDIZED MODEL RESULTS

STDYX Standardization

                                                    Two-Tailed   Rate of
                  Estimate       S.E.  Est./S.E.    P-Value    Missing

 READ9    ON
    READ1            0.701      0.044     15.767      0.000      0.102
    LRNPROB1        -0.168      0.082     -2.036      0.042      0.157
    BEHSYMP1        -0.153      0.082     -1.861      0.063      0.159

 Intercepts
    READ9            4.424      0.531      8.332      0.000      0.152

 Residual Variances
    READ9            0.394      0.055      7.166      0.000      0.099

R-SQUARE

    Observed                                        Two-Tailed   Rate of
    Variable        Estimate       S.E.  Est./S.E.    P-Value    Missing

    READ9            0.606      0.055     11.033      0.000      0.099
```

Analyzing Imputations in R

Returning to the previous Blimp script, the `SAVE` command and the `stacked` keyword saved the imputed data sets to a single stacked file with an index variable in the first column identifying the individual files. The stacked file is appropriate for analyzing data in R, SAS, SPSS, and Stata, among others.

The R input file for the analysis is `Ex6.R`. The code block below shows the commands that import the data.

R Script Ex6.R

```
1   library(fdir)
2   library(mitml)
3   set()
4   imps <- read.table("./imps/imps.dat")
5   names(imps) <- c("imputation","id","male","hispanic","riskgrp",
6     "atrisk","behsymp1","lrnprob1","read1","read2","read3",
7     "read9","read9grp","stanread7","math1","math2","math3",
8     "math9","math9grp","stanmath7")
```

The example requires the `fdir` and `lavaan` packages, which are loaded on lines 1 and 2. On line 3, the `set()` function of the `fdir` package identifies the file path of the folder containing the R script and sets this location as the working directory. On line 4, the `read.table` command imports the stacked data. It is only necessary to specify the name of the input data file. No file path is required when the data reside in the same folder as the R script as is the case here. Finally, variable names are listed beginning on line 5. Importantly, the first variable named `IMPUTATION` is the index that identifies the individual files.

The next block of code relies on the `mitml` package to fit the model to each data set and pool the results using Rubin's rules. The `implist` command on line 9 unstacks the data and creates a list that contains the individual files. Line 10 fits the focal regression model using the `lm` function, and line 11 uses the `testEstimates` function in `mitml` to implement Rubin's pooling rules and save the results in an object called `estimates`. The `df.com` parameter is the denominator degrees of freedom that would have resulted had there been no missing data (i.e., $N–K–1$ degrees of freedom, where K is the number of predictors). This argument produces Barnard and Rubin degrees of freedom values. Finally, lines 12 and 13 print the estimates and confidence intervals.

R Script Ex6.1.R, continued

```
implist <- as.mitml.list(split(imps, imps$imputation))
fit <- with(implist, lm(read9 ~ read1 + lrnprob1 + behsymp1))
estimates <- testEstimates(fit, extra.pars = T, df.com = 134)
estimates
confint(estimates)
```

When fitting regression models to complete data sets, researchers often use an omnibus *F* test to evaluate the set of slope coefficients. The `testModels` command below specifies a multiple imputation Wald *F* statistic evaluating the null hypothesis that the population slopes equal 0 (Li et al., 1991). The test requires an additional model on line 14 that represents the null hypothesis, which in this case is an empty regression model with just an intercept. On line 15, the full model and null model objects passed into the `testModels` function, and the `D1` keyword requests the Wald test. As before, the `df.com` parameter is the denominator degrees of freedom that would have resulted had there been no missing data. This argument produces the Barnard and Rubin (1999) degrees of freedom adjustment.

R Script Ex6.1.R, continued

```
null <- with(implist, lm(read9 ~ 1))
testModels(fit, null, df.com = 134, method = "D1")
```

R Output

The table of unstandardized pooled parameter estimates is shown below. The first two columns display the pooled unstandardized estimates and standard errors, and the third through fifth columns display the corresponding test statistics. The focal model results are shown in bold typeface. The `RIV` column (relative increase in variance) is a fraction comparing imputation noise to complete-data sampling variation, and the `FMI` column (fraction of missing information in the literature) quantifies the imputation noise in each estimate as proportion of its squared standard error.

```
> estimates

Call:

testEstimates(model = fit, extra.pars = T, df.com = 134)

Final parameter estimates and inferences obtained from 20 imputed data sets.

             Estimate Std.Error   t.value        df   P(>|t|)       RIV       FMI
(Intercept)    65.487     5.877    11.144   100.498     0.000     0.169     0.161
read1           0.506     0.043    11.725    92.752     0.000     0.212     0.192
lrnprob1       -0.231     0.114    -2.022   100.704     0.046     0.168     0.160
behsymp1       -0.189     0.102    -1.841    97.962     0.069     0.182     0.171

                   Estimate
Residual~~Residual   88.944

Hypothesis test adjusted for small samples with df=[134]
complete-data degrees of freedom.

> confint(estimates)
                 2.5 %      97.5 %
(Intercept) 53.8288728 77.14584684
read1        0.4202903  0.59168880
lrnprob1    -0.4581615 -0.00433096
behsymp1    -0.3919669  0.01475078
```

The results are interpreted in the same way as a complete-data regression analysis. For example, consider the first-grade reading score slope. The model predicts that two individuals who differ by one point on `READ1` but are the same on `LRNPROB1` and `BEHSYMP1` should differ by .51 points on `READ9`. The corresponding test statistic indicates that the slope coefficient is statistically different from zero ($t = 11.73$, $p < .001$). Note that these estimates are numerically identical to those from Bayesian and maximum likelihood estimation.

Finally, the Wald omnibus F statistic is shown in the output table below. The test statistic is statistically significant, thus refuting the null hypothesis that all population slopes equal zero.

```
Model comparison calculated from 20 imputed data sets.
Combination method: D1

    F.value     df1     df2   P(>F)     RIV
     58.272       3 123.487   0.000   0.177

Hypothesis test adjusted for small samples with df=[134]
complete-data degrees of freedom.
```

EXAMPLE 7: LOGISTIC REGRESSION WITH A BINARY OUTCOME

This example illustrates a binary logistic regression analysis with incomplete data. The analysis uses the `behaviorachievement.dat` data set taken from a longitudinal study that followed 138 students from primary through middle school. The file includes three annual assessments of broad reading and math achievement beginning in the first grade, seventh grade standardized achievement test scores taken from a statewide assessment, and a final measure of broad reading and math obtained in ninth grade. The data also contain teacher ratings of behavioral symptoms and learning problems were also obtained in the first grade. The data description at the beginning of this document provides additional details. The variables for this analysis are as follows.

Name	Definition	Missing %	Scale
	Focal Variables		
BEHSYMP1	1st grade behavioral symptoms	3.6	Numeric
LRNPROB1	1st grade learning problems	2.2	Numeric
READ1	1st grade broad reading composite	6.5	Numeric
READ9GRP	9th grade reading classification	17.4	0 = Below average, 1 = Average
	Auxiliary Variables		
READ2	2nd grade broad reading composite	9.4	Numeric
STANREAD7	7th grade standardized math	19.6	Numeric

Analysis Model

The analysis model features a binary classification of ninth grade reading performance regressed on first grade reading achievement and teacher-rated learning problems and behavioral symptoms.

$$\text{logit}(READ9GRP) = \beta_0 + \beta_1(READ_1) + \beta_2(LRNPROB_1) + \beta_3(BEHSYMP_1) \qquad (16)$$

Unlike a complete-data regression analysis, all incomplete variables require distributional assumptions, including the predictors. Blimp automatically assigns a multivariate normal distribution to the predictors.

The missing data literature often recommends an inclusive strategy that incorporates auxiliary variables that either predict missingness or correlate with the incomplete variables (Collins et al., 2001). Following the same factored regression specification from earlier examples, auxiliary variables enter the model as additional outcomes that are predicted by the analysis variables and by each other. The additional regression equations are as follows.

$$\begin{aligned} READ_2 &= \gamma_{01} + \gamma_{11}(READ9GRP) + \gamma_{21}(READ_1) \\ &+ \gamma_{31}(LRNPROB_1) + \gamma_{41}(BEHSYMP_1^*) + \epsilon_1 \\ STANREAD_7 &= \gamma_{02} + \gamma_{12}(READ_2) + \gamma_{22}(READ9GRP) \\ &+ \gamma_{32}(READ_1) + \gamma_{42}(LRNPROB_1) + \gamma_{52}(BEHSYMP_1^*) + \epsilon_2 \end{aligned} \tag{17}$$

Along with the logistic regression model from Equation 16, the collection of regressions can be viewed as a path model, where the focal regression is one part of a larger network (see the path diagram from Example 2). The key difference is that the path coefficients are just a tool for linking incomplete variables and do not represent a substantive theory.

Blimp Script

The code block below shows Blimp script `Ex7.1.inp`. The first six lines can be viewed as a set of commands that specify information about the data and variables. The `DATA` command specifies the name of the input text file. No file path is required when the data file is located in the same directory as the script, as it is here. Starting on line 2, the `VARIABLES` command names the data columns. The `ORDINAL` command on line 5 defines the outcome as categorical. Binary variables can be defined as ordinal or nominal, as the statistical models are identical. The `MISSING` command on line 6 defines a global missing value code as 999.

Blimp Script Ex7.1.imp

```
1   DATA: behaviorachievement.dat;
2   VARIABLES: id male hispanic riskgrp atrisk behsymp1 lrnprob1
3     read1 read2 read3 read9 read9grp stanread7
4     math1 math2 math3 math9 math9grp stanmath7;
5   ORDINAL: read9grp;
6   MISSING: 999;
7   MODEL:
8   focal.model:
9   logit(read9grp) ~ read1@beta1 lrnprob1@beta2 behsymp1@beta3;
10  auxiliary.models:
11  stanread7 read2 ~ read9grp read1 lrnprob1 behsymp1;
12  TEST:
13  beta1:beta3 = 0;
14  SEED: 90291;
15  BURN: 1000;
16  ITERATIONS: 10000;
```

The `MODEL` and `TEST` blocks can be viewed as a set. The `MODEL` command lists the regression models, with outcome variables to the left of the tilde and predictors to the right. The code uses labels (`focal.model` and `auxiliary.models`) to order output tables, such that the logistic model appears first followed by the auxiliary variable models. The focal model listed on line 9 assigns labels the slope coefficients using the @ symbol. Listing the dependent variable inside the `logit` function triggers logistic regression rather than the default probit regression. Blimp automatically configures the explanatory variable models under the assumption that they are normally distributed. Line 11 is a syntax shortcut that produces the two auxiliary variable regression models in Equation 17; in the first model, `READ2` is regressed on the focal variables, and the second model features `STANREAD7` regressed on `READ2` and the focal variables. The `TEST` command uses the parameter labels to specify a custom hypothesis test that all three slopes equal 0. This command produces the Bayesian Wald test (Asparouhov & Muthén, 2021), which is essentially a chi-square statistic that captures the discrepancy between the Bayesian point estimates (posterior means) and the hypothesized values of zero.

Finally, lines 14 through 16 can be viewed as a block of commands that specify features of the MCMC algorithm: the `SEED` command gives an integer string that initializes the random number generator, the `BURN` command specifies the number of iterations for the warm-up or burn-in

period, and the `ITERATIONS` command gives the number of MCMC iterations on which the analysis summaries are based (essentially, the number of MCMC cycles following the warm-up period).

Blimp Output

Prior to inspecting the parameter estimates, it is important to investigate the potential scale reduction (PSR) factor diagnostics (Gelman & Rubin, 1992) to determine whether MCMC has converged. Blimp divides the burn-in period into 20 equal segments, and it computes the PSR diagnostic for every parameter. The table located near the top of the output reports the highest (worst) PSR value across all parameters in every model. A common recommendation is that these values should be less than 1.05 or perhaps 1.10 (Asparouhov & Muthén, 2010a; Gelman et al., 2014). If the PSR in the bottom row of the table (the final check of the burn-in period) is above these cutoffs, then rerun the analysis with a longer burn-in period.

```
BURN-IN POTENTIAL SCALE REDUCTION (PSR) OUTPUT:

  NOTE: Split chain PSR is being used. This splits each chain's
        iterations to create twice as many chains.

  Comparing iterations across 2 chains     Highest PSR   Parameter #
                            26 to 50             1.140             2
                            51 to 100            1.072             2
                            76 to 150            1.041             3
                           101 to 200            1.041            37
                           126 to 250            1.033             6
                           151 to 300            1.030             6
                           176 to 350            1.028            37
                           201 to 400            1.023            37
                                  ...              ...            ..
                           401 to 800            1.008            37
                           426 to 850            1.009            37
                           451 to 900            1.009            37
                           476 to 950            1.008            19
                           501 to 1000           1.008            37
```

The next section of the output displays information about the variables in the analysis and the models used for estimation. This output table mimics the one from Example 6. In the interest of space, we point readers to that example for additional details.

Most software programs that fit regression models report an omnibus *F* test that evaluates the set of slope coefficients. The `TEST` command in the previous script requested an analogous Bayesian Wald chi-square statistic (Asparouhov & Muthén, 2021) that evaluates the null hypothesis that all population slopes equal zero. The chi-square statistic, degrees of freedom, and *p*-value appear near the bottom of the `MODEL FIT` section under the `WALD TEST` heading. The test statistic is statistically significant, thus refuting the null hypothesis.

```
MODEL FIT:

  ...

  WALD TESTS (Asparouhov & Muthén, 2021)

  Test #1

    Full:
      [1]  logit(read9grp) ~ Intercept read1@beta1 lrnprob1@beta2 behsymp1@beta3

    Restricted:
      [1]  logit(read9grp) ~ Intercept read1@beta1 lrnprob1@beta2 behsymp1@beta3

    Constraints in Restricted:
      [1]  beta1 = 0
      [2]  beta2 = 0
      [3]  beta3 = 0

    Wald Statistic (Chi-Square)                24.106
    Number of Parameters Tested (df)                3
    Probability                                 0.000
```

The table summarizing the focal regression model is shown below. The table includes unstandardized coefficients, standardized slopes, and variance explained effect size estimates.

```
OUTCOME MODEL ESTIMATES:

  Summaries based on 10000 iterations using 2 chains.

  focal.model block:

Outcome Variable:  logit(read9grp)

Parameters                          Median     StdDev       2.5%      97.5%        PSR      N_Eff
                                ------------------------------------------------------------------

Coefficients:
  Intercept                         -2.740      1.284     -5.289     -0.241      1.006   5231.685
  read1                              0.062      0.013      0.038      0.089      1.010   3171.888
  lrnprob1                          -0.034      0.031     -0.096      0.024      1.005   3810.076
  behsymp1                          -0.022      0.026     -0.073      0.028      1.005   4367.404

Odds Ratio:
  Intercept                          0.065      0.284      0.005      0.786      1.003   6917.301
  read1                              1.064      0.014      1.039      1.094      1.010   3172.490
  lrnprob1                           0.967      0.029      0.908      1.024      1.005   3807.143
  behsymp1                           0.978      0.025      0.929      1.029      1.005   4366.400

Proportion Variance Explained
  by Coefficients                    0.154      0.057      0.068      0.287      1.008   5059.652
  by Residual Variation              0.846      0.057      0.713      0.932      1.008   5059.652
                                ------------------------------------------------------------------
```

MCMC estimation produces a distribution for each parameter in the table. The median and standard deviation columns describe the center and spread of the posterior distributions; although they make no reference to drawing repeated samples, they are analogous—and numerically equivalent in most cases—to frequentist point estimates and standard errors. The 95% credible intervals in the rightmost columns give a range that captures 95% of the parameter's distribution. These are akin to confidence intervals, but the intervals describe parameter distributions rather than characteristics of repeated samples. The `N_Eff` values in rightmost column of the table give the effective number of MCMC samples for each parameter. These quantities essentially represent the number of independent estimates on which the parameter summaries are based after removing autocorrelations from the MCMC process. Gelman et al. (2014, p. 287) recommend values greater than 100. All values in the example table exceed this recommended minimum. In cases where the `N_Eff` values are insufficient, increasing the value on the `ITERATIONS` command will remedy the issue.

The results are interpreted in the same way as a complete-data logistic regression analysis. For example, consider the first-grade reading score slope. The model predicts that the logits for two

individuals who differ by one point on READ1 but are the same on LRNPROB1 and BEHSYMP1 differ by 0.07. The 95% credible interval limits suggest this effect is statistically different from zero ($p < .05$) because the null value is well outside the interval. The printed output also includes the table of odds ratios that reflect multiplicative changes to the odds. For example, a one-point increase in first grade reading scores increases the odds of average or higher ninth grade reading by a factor 1.07, holding first grade learning problems and behavioral symptoms constant. Collectively, the predictors explain 17% of the variation in the underlying logistic latent variable. Note that the tabled values are numerically identical to the maximum likelihood estimates from Example 2.

The Blimp output also includes tables of regression model parameters for the auxiliary variables as well as the auto-generated models for incomplete predictors. These additionally results are not of substantive interest and would not be reported. The auxiliary variable models appear in OUTCOME MODEL ESTIMATES section with the focal results, and the auto-generated predictor models are displayed under the heading PREDICTOR MODEL ESTIMATES.

Saving Multiple Imputations

MCMC estimation imputes missing values at every iteration, such that the resulting Bayesian estimates average over thousands of plausible replacement scores (10,000 sets in this example). A subset of the imputations can be saved for reanalysis in the frequentist framework, if desired. The Blimp input file Ex7.2.imp is identical Ex7.1.imp, but it adds the following lines at the bottom of the script.

```
NIMPS: 20;
CHAINS: 20;
SAVE:
stacked = ./imps/imps.dat;
separate = ./imps/imp*.dat;
```

The NIMPS, CHAINS and SAVE commands can be viewed as a set. Setting NIMPS equal to CHAINS saves a single filled-in data set from the final iteration of a unique MCMC process, thus avoiding autocorrelation among the imputations. The SAVE command provides a name for the imputed data sets. The script illustrates how to save data sets in two common formats. The stacked keyword creates a stacked file where all imputations are in a single file, and the separate keyword saves each imputed data set to a separate file with the asterisk replaced by a numeric

index. To keep things organized, the `./imps` part of the file path points to a subfolder named `imps` located within the same folder as the script and data. The separate keyword also creates a list of file names needed for analysis in Mplus (in this example, a filed called `implist.dat` located in the `imps` folder).

When saving imputations, the bottom of the Blimp output file displays a table listing the order of the variables in the output data sets. All variables are saved regardless of whether they appeared in the fitted models. When saving data to a stacked file (e.g., for analysis in R or other packages), the first variable in the file is an integer index that identifies which data set each row belongs to (e.g., an integer variable that ranges from 1 to 20 in this example).

```
VARIABLE ORDER IN IMPUTED DATA:

separate = './imps/imp*.dat'

   id male hispanic riskgrp atrisk behsymp1 lrnprob1 read1 read2 read3
  read9 read9grp stanread7 math1 math2 math3 math9 math9grp stanmath7

stacked = './imps/imps.dat'

   imp# id male hispanic riskgrp atrisk behsymp1 lrnprob1 read1 read2 read3
   read9 read9grp stanread7 math1 math2 math3 math9 math9grp stanmath7
```

The imputed data sets are subsequently analyzed in another software package, and estimates and standard errors are combined using Rubin's rules (Little & Rubin, 2020). The analysis phase does not utilize the auxiliary variables, as their information is embedded in the imputations. Scripts for analyzing the imputed data sets are found in the next subsections.

Analyzing Imputations in Mplus

In lieu of the Bayesian estimates, Blimp's `SAVE` command can be used to save multiple imputations for analysis in the frequentist framework. Returning to the previous Blimp script, the `SAVE` command and the `separate` keyword saved each imputed data set to a separate file with the asterisk replaced by a numeric index. The separate keyword also creates a list of file names needed for analysis in Mplus (in this example, a filed called `implist.dat` located in the `imps` subfolder). Example 6 shows the contents of this file.

The Mplus input file for analyzing the imputations is `Ex7.inp`. The script is similar to the `Ex2.1.inp` file described in Example 2 with three exceptions. First, instead of naming the raw data set, the `DATA` command lists the text file containing the names of the imputed data sets (the `implist.dat` file located in the `./imps` subdirectory). The `type = imputation` subcommand instructs Mplus that the input data is a list of file names. Second, the missing subcommand is omitted because the analysis variables are now complete. Finally, the `MODEL` section no longer specifies a normal distribution for the predictors or models for the auxiliary variables. Readers can refer back to Example 2 for a detailed description of the other commands. The code block below shows the analysis and pooling script.

Mplus Script Ex7.inp

```
DATA:
file = ./imps/implist.dat;
type = imputation;
VARIABLE:
names = id male hispanic riskgrp atrisk behsymp1 lrnprob1
  read1 read2 read3 read9 read9grp stanread7
  math1 math2 math3 math9 math9grp stanmath7;
usevariables = read9grp read1 lrnprob1 behsymp1;
categorical = read9grp;
ANALYSIS:
estimator = ml;
link = logit;
MODEL:
read9grp on read1 lrnprob1 behsymp1 (beta1-beta3);
MODEL TEST:
0 = beta1; 0 = beta2; 0 = beta3;
OUTPUT:
stdyx cinterval;
```

Mplus Output

When fitting regression models to complete data sets, researchers often use an omnibus F test to evaluate the set of slope coefficients. The `MODEL TEST` command specified a multiple imputation Wald chi-square statistic evaluating the null hypothesis that the population slopes equal 0 (Asparouhov & Muthén, 2010b). The chi-square statistic, degrees of freedom, and p-value appear

near the bottom of the `MODEL FIT INFORMATION` section under the `Wald Test of Parameter Constraints` heading. The test statistic is statistically significant, thus refuting the null hypothesis.

```
MODEL FIT INFORMATION

Number of Free Parameters                        4

...

Wald Test of Parameter Constraints

          Value                            23.342
          Degrees of Freedom                    3
          P-Value                           0.0001
```

The table of unstandardized parameter estimates is shown below. The first two columns display the pooled unstandardized estimates and standard errors, and the third and fourth columns display the corresponding *z*-statistics and *p*-values. The focal model results are shown in bold typeface. The `Rate of Missing` column (also called the fraction of missing information in the literature) quantifies the imputation noise in each estimate as proportion of its squared standard error.

```
MODEL RESULTS
                                                    Two-Tailed   Rate of
                   Estimate       S.E.  Est./S.E.    P-Value    Missing

READ9GRP   ON
    READ1             0.068      0.015      4.463      0.000      0.173
    LRNPROB1         -0.029      0.030     -0.971      0.331      0.188
    BEHSYMP1         -0.019      0.025     -0.762      0.446      0.121
 Thresholds
    READ9GRP$1        3.602      1.672      2.154      0.031      0.231
```

The results are interpreted in the same way as a complete-data logistic regression analysis. For example, consider the first-grade reading score slope. The model predicts that the logits for two individuals who differ by one point on `READ1` but are the same on `LRNPROB1` and `BEHSYMP1` differ by 0.07. The corresponding test statistic indicates that the slope coefficient is statistically different from zero ($z = 4.46$, $p < .001$). Note that Mplus reports a threshold parameter instead of the usual regression intercept. The threshold from a binary logistic model has the same value but opposite sign as the intercept (i.e., $\hat{\beta}_0 = -3.60$). Note that these estimates are numerically equivalent to those from Bayesian and maximum likelihood estimation.

Finally, the printed output also includes the table of odds ratios that reflect multiplicative changes to the odds. For example, a one-point increase in first grade reading scores increases the odds of average or higher ninth grade reading by a factor 1.08, holding first grade learning problems and behavioral symptoms constant.

```
CONFIDENCE INTERVALS OF MODEL RESULTS

                 Lower 2.5%    Lower 5%    Estimate    Upper 5%  Upper 2.5%%

READ9GRP ON
    READ1             0.038       0.043       0.068       0.092       0.097
    LRNPROB1         -0.089      -0.079      -0.029       0.020       0.030
    BEHSYMP1         -0.068      -0.060      -0.019       0.022       0.030

 Thresholds
    READ9GRP$1        0.324       0.851       3.602       6.352       6.879

CONFIDENCE INTERVALS FOR THE LOGISTIC REGRESSION ODDS RATIO RESULTS

 READ9GRP ON
    READ1             1.039       1.044       1.070       1.097       1.102
    LRNPROB1          0.915       0.924       0.971       1.021       1.030
    BEHSYMP1          0.934       0.942       0.981       1.022       1.030
```

Analyzing Imputations in R

Returning to the previous Blimp script, the `SAVE` command and the `stacked` keyword saved the imputed data sets to a single stacked file with an index variable in the first column identifying the

individual files. The stacked file is appropriate for analyzing data in R, SAS, SPSS, and Stata, among others.

The R input file for the analysis is `Ex7.R`. The code block below shows the commands that import the data.

R Script Ex7.R

```
1   library(fdir)
2   library(mitml)
3   set()
4   imps <- read.table("./imps/imps.dat")
5   names(imps) <- c("imputation","id","male","hispanic","riskgrp",
6     "atrisk","behsymp1","lrnprob1","read1","read2","read3",
7     "read9","read9grp","stanread7","math1","math2","math3",
8     "math9","math9grp","stanmath7")
```

The example requires the `fdir` and `lavaan` packages, which are loaded on lines 1 and 2. On line 3, the `set()` function of the `fdir` package identifies the file path of the folder containing the R script and sets this location as the working directory. On line 4, the `read.table` command imports the stacked data. It is only necessary to specify the name of the input data file. No file path is required when the data reside in the same folder as the R script as is the case here. Finally, variable names are listed beginning on line 5. Importantly, the first variable named `IMPUTATION` is the index that identifies the individual files.

The next block of code relies on the `mitml` package to fit the model to each data set and pool the results using Rubin's rules. The `implist` command on line 9 unstacks the data and creates a list that contains the individual files. Line 10 fits the focal regression model using the `glm` function, and line 12 uses the `testEstimates` function in `mitml` to implement Rubin's pooling rules and save the results in an object called `estimates`. The `df.com` parameter is the denominator degrees of freedom that would have resulted had there been no missing data (i.e., $N–K–1$ degrees of freedom, where K is the number of predictors). This argument produces Barnard and Rubin degrees of freedom values. Finally, lines 13 and 14 print the estimates and confidence intervals.

R Script Ex7.1.R, continued

```
implist <- as.mitml.list(split(imps, imps$imputation))
fit <- with(implist, glm(read9grp ~ read1 + lrnprob1 + behsymp1,
   family = "binomial"))
estimates <- testEstimates(fit, extra.pars = T, df.com = 134)
estimates
confint(estimates)
```

When fitting regression models to complete data sets, researchers often use an omnibus *F* test to evaluate the set of slope coefficients. The `testModels` command below specifies a multiple imputation Wald *F* statistic evaluating the null hypothesis that the population slopes equal 0 (Li et al., 1991). The test requires an additional model on line 15 that represents the null hypothesis, which in this case is an empty regression model with just an intercept. On line 15, the full model and null model objects passed into the `testModels` function, and the `D1` keyword requests the Wald test. As before, the `df.com` parameter is the denominator degrees of freedom that would have resulted had there been no missing data. This argument produces the Barnard and Rubin (1999) degrees of freedom adjustment.

R Script Ex7.1.R, continued

```
null <- with(implist, glm(read9grp ~ 1, family = "binomial"))
testModels(fit, null, df.com = 134, method = "D1")
```

R Output

The table of unstandardized parameter estimates is shown below. The first two columns display the pooled unstandardized estimates and standard errors, and the third through fifth columns display the corresponding test statistics. The focal model results are shown in bold typeface. The `RIV` column (relative increase in variance) is a fraction comparing imputation noise to complete-data sampling variation, and the `FMI` column (fraction of missing information in the literature) quantifies the imputation noise in each estimate as proportion of its squared standard error.

```
> estimates

Call:

testEstimates(model = analysis, df.com = 134)

Final parameter estimates and inferences obtained from 20 imputed data sets.

             Estimate Std.Error   t.value        df   P(>|t|)       RIV       FMI
(Intercept)    -3.602     1.672    -2.154    79.914     0.034     0.294     0.246
read1           0.068     0.015     4.463    93.795     0.000     0.206     0.188
lrnprob1       -0.029     0.030    -0.971    90.209     0.334     0.227     0.202
behsymp1       -0.019     0.025    -0.762   106.973     0.448     0.135     0.135

Hypothesis test adjusted for small samples with df=[134]
complete-data degrees of freedom.

> confint(estimates)
                  2.5 %      97.5 %
(Intercept) -6.92951753 -0.27356800
read1        0.03751055  0.09763918
lrnprob1    -0.08980267  0.03082041
behsymp1    -0.06832451  0.03038596
```

The results are interpreted in the same way as a complete-data logistic regression analysis. For example, consider the first-grade reading score slope. The model predicts that the logits for two individuals who differ by one point on `READ1` but are the same on `LRNPROB1` and `BEHSYMP1` differ by 0.07. The corresponding test statistic indicates that the slope coefficient is statistically different from zero ($t = 4.46$, $p < .001$). Note that these estimates are numerically identical to those from Bayesian and maximum likelihood estimation.

Finally, the Wald omnibus F statistic is shown in the output table below. The test statistic is statistically significant, thus refuting the null hypothesis that all population slopes equal 0.

```
Model comparison calculated from 20 imputed data sets.
Combination method: D1

    F.value     df1     df2   P(>F)     RIV
      7.359       3 120.633   0.000   0.214

Hypothesis test adjusted for small samples with df=[134]
complete-data degrees of freedom.
```

EXAMPLE 8: REGRESSION WITH BINARY AND ORDINAL PREDICTORS

This example illustrates a multiple regression analysis with incomplete categorical predictors. The analysis uses the `mathachievement.dat` data set taken from an educational intervention where 250 students were assigned to an intervention and comparison condition. The file includes pretest and posttest math achievement scores, a measure of math self-efficacy, standardized reading scores taken from a statewide assessment, and several sociodemographic variables. The analysis variables are as follows.

Name	Definition	Missing %	Scale
	Focal Variables		
MATHPOST	Math achievement posttest	18.0	Numeric
CONDITION	Experimental condition	0	0 = Comparison, 1 = Intervention
FRLUNCH	Lunch assistance code	4.4	0 = None, 1 = Free/reduced lunch
EFFICACY	Math self-efficacy rating	9.6	Ordinal (1 to 6)
MATHPRE	Math achievement pretest	0	Numeric
	Auxiliary Variables		
ATRISK	Behavioral disorder risk	5.2	0 = Low risk, 1 = At-risk
STANREAD	Standardized reading	9.2	Numeric

Analysis Model

The analysis model features math posttest scores regressed on the experimental condition and lunch assistance dummy codes, math self-efficacy ratings, and math pretest scores.

$$\begin{aligned} MATHPOST = \beta_0 + \beta_1(CONDITION) + \beta_2(FRLUNCH) \\ + \beta_3(EFFICACY) + \beta_4(MATHPRE) + \varepsilon \end{aligned} \tag{18}$$

Unlike a complete-data regression analysis, all incomplete variables require distributional assumptions, including the predictors. In this case, the predictor set includes incomplete binary and ordinal variables. Blimp uses a probit regression formulation that envisions discrete responses as arising from underlying continuous latent response variables. The software assumes that continuous predictors and the latent response variables are multivariate normal.

The missing data literature often recommends an inclusive strategy that incorporates auxiliary variables that either predict missingness or correlate with the incomplete variables (Collins et al., 2001). Following earlier examples, auxiliary variables enter the model as additional outcomes that are predicted by the analysis variables and by each other. The additional regression equations are as follows.

$$\begin{aligned} ATRISK^{*} &= \gamma_{01} + \gamma_{11}(MATHPOST) + \gamma_{21}(CONDITION) \\ &+ \gamma_{31}(FRLUNCH) + \gamma_{41}(EFFICACY) + \gamma_{51}(MATHPRE) + \epsilon_1 \\ STANREAD &= \gamma_{02} + \gamma_{12}(ATRISK) + \gamma_{22}(MATHPOST) + \gamma_{32}(CONDITION) \\ &+ \gamma_{42}(FRLUNCH) + \gamma_{52}(EFFICACY) + \gamma_{62}(MATHPRE) + \epsilon_2 \end{aligned} \tag{19}$$

The `ATRISK` model is a probit regression, with the binary outcome model as a latent response variable (denoted by the asterisk superscript). Again, the entire collection of regressions can be viewed as a path model, where the focal regression is one part of a larger network (see the path diagram from Example 2). The key difference is that the path coefficients are just a tool for linking incomplete variables and do not represent a substantive theory.

Blimp Script

The code block below shows Blimp script `Ex8.1.inp`. The first five lines can be viewed as a set of commands that specify information about the data and variables. The `DATA` command specifies the name of the input text file. No file path is required when the data file is located in the same directory as the script, as it is here. Starting on line 2, the `VARIABLES` command names the data columns. The `ORDINAL` command on line 4 identifies binary and ordinal variables. Binary variables can be defined as ordinal or nominal, as the statistical models are identical. The `MISSING` command on line 5 defines a global missing value code as 999.

Blimp Script Ex8.1.imp

```
1  DATA: mathachievement.dat;
2  VARIABLES: id condition male frlunch atrisk stanread efficacy anxiety
3     mathpre mathpost;
4  ORDINAL: condition frlunch atrisk efficacy;
5  MISSING: 999;
6  FIXED: condition mathpre;
7  MODEL:
8  focal.model:
9  mathpost ~ condition@beta1 frlunch@beta2 efficacy@beta3 mathpre@beta4;
10 auxiliary.models:
11 stanread atrisk ~ mathpost condition frlunch efficacy mathpre;
12 TEST:
13 beta1:beta4 = 0;
14 SEED: 90291;
15 BURN: 5000;
16 ITERATIONS: 10000;
```

The `FIXED`, `MODEL`, and `TEST` blocks can be viewed as a set. The `FIXED` command identifies the two complete variables, which do not require a distribution or regression model. Beginning on line 7, the `MODEL` command lists the regression models, with outcome variables to the left of the tilde and predictors to the right. The code uses labels (`focal.model` and `auxiliary.models`) to order output tables, such that the focal model appears first followed by the auxiliary variable models. The focal model listed on line 9 assigns labels the slope coefficients using the @ symbol. Blimp automatically configures the explanatory variable models under the assumption that they are normally distributed. Line 11 is a syntax shortcut that produces the two auxiliary variable regression models in Equation 19; in the first model, `READ2` is regressed on the focal variables, and the second model features `STANREAD7` regressed on `READ2` and the focal variables. The `TEST` command uses the parameter labels to specify a custom hypothesis test that all three slopes equal zero. This command produces the Bayesian Wald test (Asparouhov & Muthén, 2021), which is essentially a chi-square statistic that captures the discrepancy between the Bayesian point estimates (posterior means) and the hypothesized values of zero.

Finally, lines 14 through 16 can be viewed as a block of commands that specify features of the MCMC algorithm: the `SEED` command gives an integer string that initializes the random number generator, the `BURN` command specifies the number of iterations for the warm-up or burn-in

period, and the `ITERATIONS` command gives the number of MCMC iterations on which the analysis summaries are based (essentially, the number of MCMC cycles following the warm-up period).

Blimp Output

Prior to inspecting the parameter estimates, it is important to investigate the potential scale reduction (PSR) factor diagnostics (Gelman & Rubin, 1992) to determine whether MCMC has converged. Blimp divides the burn-in period into 20 equal segments, and it computes the PSR diagnostic for every parameter. The table located near the top of the output reports the highest (worst) PSR value across all parameters in every model. A common recommendation is that these values should be less than 1.05 or perhaps 1.10 (Asparouhov & Muthén, 2010a; Gelman et al., 2014). If the PSR in the bottom row of the table (the final check of the burn-in period) is above these cutoffs, then rerun the analysis with a longer burn-in period.

```
BURN-IN POTENTIAL SCALE REDUCTION (PSR) OUTPUT:

  NOTE: Split chain PSR is being used. This splits each chain's
        iterations to create twice as many chains.

  Comparing iterations across 2 chains     Highest PSR   Parameter #
                           126 to 250           1.416            59
                           251 to 500           1.425            57
                           376 to 750           1.146            57
                           501 to 1000          1.303            58
                           626 to 1250          1.073            58
                           751 to 1500          1.068            59
                           876 to 1750          1.097            59
                          1001 to 2000          1.052            59
                                   ...            ...            ..
                          2001 to 4000          1.022            56
                          2126 to 4250          1.035            57
                          2251 to 4500          1.040            57
                          2376 to 4750          1.016            56
                          2501 to 5000          1.009            56
```

The next section of the output displays information about the variables in the analysis and the models used for estimation. This output table mimics the one from Example 6. In the interest of space, we point readers to that example for additional details.

Most software programs that fit regression models report an omnibus *F* test that evaluates the set of slope coefficients. The `TEST` command in the previous script requested an analogous Bayesian Wald chi-square statistic (Asparouhov & Muthén, 2021) that evaluates the null hypothesis that all population slopes equal zero. The chi-square statistic, degrees of freedom, and *p*-value appear near the bottom of the `MODEL FIT` section under the `WALD TEST` heading. The test statistic is statistically significant, thus refuting the null hypothesis.

```
MODEL FIT:

  ...

  WALD TESTS (Asparouhov & Muthén, 2021)

  Test #1

    Full:
      [1]  mathpost ~ Intercept condition@beta1 frlunch@beta2 efficacy@beta3
                   mathpre@beta4

    Restricted:
      [1]  mathpost ~ Intercept condition@beta1 frlunch@beta2 efficacy@beta3
                   mathpre@beta4

    Constraints in Restricted:
      [1]  beta1 = 0
      [2]  beta2 = 0
      [3]  beta3 = 0
      [4]  beta4 = 0

    Wald Statistic (Chi-Square)              142.310
    Number of Parameters Tested (df)               4
    Probability                                0.000
```

The table summarizing the focal regression model is shown below. The table includes unstandardized coefficients, standardized slopes, and variance explained effect size estimates.

```
OUTCOME MODEL ESTIMATES:

  Summaries based on 10000 iterations using 2 chains.

  focal.model block:

Outcome Variable:  mathpost

Parameters                      Median     StdDev       2.5%      97.5%        PSR      N_Eff
                              ------------------------------------------------------------------
Variances:
  Residual Var.                 53.303      5.509     43.695     65.373      1.000   5636.603

Coefficients:
  Intercept                     28.345      3.088     22.308     34.520      1.000   7012.232
  condition                      2.263      1.047      0.202      4.312      1.000   6953.241
  frlunch                       -5.502      1.095     -7.608     -3.324      1.000   5344.564
  efficacy                       0.831      0.346      0.160      1.517      1.000   4712.974
  mathpre                        0.530      0.062      0.408      0.653      1.000   6537.163

Standardized Coefficients:
  condition                      0.117      0.054      0.011      0.222      1.000   6897.665
  frlunch                       -0.281      0.052     -0.378     -0.173      1.000   5652.427
  efficacy                       0.139      0.057      0.027      0.248      1.000   4706.081
  mathpre                        0.477      0.048      0.378      0.564      1.000   6400.853

Proportion Variance Explained
  by Coefficients                0.426      0.046      0.328      0.510      1.000   5954.279
  by Residual Variation          0.574      0.046      0.490      0.672      1.000   5954.279

                              ------------------------------------------------------------------
Proportion Variance Explained
  by Coefficients                0.424      0.046      0.328      0.508      1.002   6225.855
  by Residual Variation          0.576      0.046      0.492      0.672      1.002   6225.855

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

MCMC estimation produces a distribution for each parameter in the table. The median and standard deviation columns describe the center and spread of the posterior distributions; although they make no reference to drawing repeated samples, they are analogous—and numerically equivalent in most cases—to frequentist point estimates and standard errors. The 95% credible intervals in the rightmost columns give a range that captures 95% of the parameter's distribution. These are akin to confidence intervals, but the intervals describe parameter distributions rather than characteristics of repeated samples. The `N_Eff` values in

rightmost column of the table give the effective number of MCMC samples for each parameter. These quantities essentially represent the number of independent estimates on which the parameter summaries are based after removing autocorrelations from the MCMC process. Gelman et al. (2014, p. 287) recommend values greater than 100. All values in the example table exceed this recommended minimum. In cases where the `N_Eff` values are insufficient, increasing the value on the `ITERATIONS` command will remedy the issue.

The results are interpreted in the same way as a complete-data regression analysis with categorical predictors. For example, consider the slope for the treatment assignment dummy code. The positive coefficient indicates that, for two students who share the same covariate profile (i.e., lunch assistance, self-efficacy, and pretest scores), the model predicts that the student in the experimental condition should score 2.26 points higher than the student in the control group. The 95% credible interval limits suggest this effect is statistically different from zero ($p < .05$) because the null value is well outside the interval. Note that the tabled values are numerically identical to the maximum likelihood estimates from Example 3.

The Blimp output also includes tables of regression model parameters for the auxiliary variables as well as the auto-generated models for incomplete predictors. These additionally results are not of substantive interest and would not be reported. The auxiliary variable models appear in `OUTCOME MODEL ESTIMATES` section with the focal results, and the auto-generated predictor models are displayed under the heading `PREDICTOR MODEL ESTIMATES`.

Saving Multiple Imputations

MCMC estimation imputes missing values at every iteration, such that the resulting Bayesian estimates average over thousands of plausible replacement scores (10,000 sets in this example). A subset of the imputations can be saved for reanalysis in the frequentist framework, if desired. The Blimp input file `Ex8.2.imp` is identical `Ex8.1.imp`, but it adds the following lines at the bottom of the script.

```
NIMPS: 20;
CHAINS: 20;
SAVE:
stacked = ./imps/imps.dat;
separate = ./imps/imp*.dat;
```

The NIMPS, CHAINS and SAVE commands can be viewed as a set. Setting NIMPS equal to CHAINS saves a single filled-in data set from the final iteration of a unique MCMC process, thus avoiding autocorrelation among the imputations. The SAVE command provides a name for the imputed data sets. The script illustrates how to save data sets in two common formats. The stacked keyword creates a stacked file where all imputations are in a single file, and the separate keyword saves each imputed data set to a separate file with the asterisk replaced by a numeric index. To keep things organized, the ./imps part of the file path points to a subfolder named imps located within the same folder as the script and data. The separate keyword also creates a list of file names needed for analysis in Mplus (in this example, a filed called implist.dat located in the imps folder).

When saving imputations, the bottom of the Blimp output file displays a table listing the order of the variables in the output data sets. All variables are saved regardless of whether they appeared in the fitted models. When saving data to a stacked file (e.g., for analysis in R or other packages), the first variable in the file is an integer index that identifies which data set each row belongs to (e.g., an integer variable that ranges from 1 to 20 in this example).

```
VARIABLE ORDER IN IMPUTED DATA:

  separate = './imps/imp*.dat'

     id condition male frlunch atrisk stanread efficacy anxiety
     mathpre mathpost

   stacked = './imps/imps.dat'

     imp# id condition male frlunch atrisk stanread efficacy
     anxiety mathpre mathpost
```

The imputed data sets are subsequently analyzed in another software package, and estimates and standard errors are combined using Rubin's rules (Little & Rubin, 2020). The analysis phase does not utilize the auxiliary variables, as their information is embedded in the imputations. Scripts for analyzing the imputed data sets are found in the next subsections.

Analyzing Imputations in Mplus

In lieu of the Bayesian estimates, Blimp's `SAVE` command can be used to save multiple imputations for analysis in the frequentist framework. Returning to the previous Blimp script, the `SAVE` command and the `separate` keyword saved each imputed data set to a separate file with the asterisk replaced by a numeric index. The separate keyword also creates a list of file names needed for analysis in Mplus (in this example, a filed called `implist.dat` located in the `imps` subfolder). Example 6 shows the contents of this file.

The Mplus input file for analyzing the imputations is `Ex8.inp`. The script is similar to the `Ex3.inp` file described in Example 3 with three exceptions. First, instead of naming the raw data set, the `DATA` command lists the text file containing the names of the imputed data sets (the `implist.dat` file located in the `./imps` subdirectory). The `type = imputation` subcommand instructs Mplus that the input data is a list of file names. Second, the missing subcommand is omitted because the analysis variables are now complete. Finally, the `MODEL` section no longer specifies a normal distribution for the predictors or models for the auxiliary variables. Readers can refer back to Example 3 for a detailed description of the other commands. The code block below shows the analysis and pooling script.

Mplus Script Ex8.inp

```
DATA:
file = ./imps/implist.dat;
type = imputation;
VARIABLE:
names = id condition male frlunch atrisk stanread
  efficacy anxiety mathpre mathpost;
usevariables = mathpost condition frlunch efficacy mathpre;
MODEL:
mathpost on condition frlunch efficacy mathpre (beta1-beta4);
MODEL TEST:
0 = beta1; 0 = beta2; 0 = beta3; 0 = beta4;
OUTPUT:
stdyx cinterval;
```

Mplus Output

When fitting regression models to complete data sets, researchers often use an omnibus *F* test to evaluate the set of slope coefficients. The `MODEL TEST` command specified a multiple imputation Wald chi-square statistic evaluating the null hypothesis that the population slopes equal 0 (Asparouhov & Muthén, 2010b). The chi-square statistic, degrees of freedom, and *p*-value appear near the bottom of the `MODEL FIT INFORMATION` section under the `Wald Test of Parameter Constraints` heading. The test statistic is statistically significant, thus refuting the null hypothesis.

```
MODEL FIT INFORMATION

Number of Free Parameters                        6

...

Wald Test of Parameter Constraints

          Value                            125.646
          Degrees of Freedom                     4
          P-Value                           0.0000
```

The table of unstandardized parameter estimates is shown below. The first two columns display the pooled unstandardized estimates and standard errors, and the third and fourth columns display the corresponding *z*-statistics and *p*-values. The focal model results are shown in bold typeface. The `Rate of Missing` column (also called the fraction of missing information in the literature) quantifies the imputation noise in each estimate as proportion of its squared standard error.

```
MODEL RESULTS

                                                    Two-Tailed   Rate of
                  Estimate       S.E.  Est./S.E.    P-Value    Missing

 MATHPOST ON
    CONDITION        2.206      1.047      2.107      0.035      0.215
    FRLUNCH         -5.392      1.086     -4.965      0.000      0.267
    EFFICACY         0.832      0.353      2.356      0.018      0.285
    MATHPRE          0.532      0.062      8.515      0.000      0.226

Intercepts
    MATHPOST        28.301      3.158      8.962      0.000      0.233

 Residual Variances
    MATHPOST        52.070      5.500      9.467      0.000      0.287
```

The results are interpreted in the same way as a complete-data regression analysis with categorical predictors. For example, consider the slope for the treatment assignment dummy code. The positive coefficient indicates that, for two students who share the same covariate profile (i.e., lunch assistance, self-efficacy, and pretest scores), the model predicts that the student in the experimental condition should score 2.21 points higher than the student in the control group. The corresponding test statistic indicates that the slope coefficient is statistically different from zero (z = 2.11, p = .04). Note that these estimates are virtually identical to those from Bayesian and maximum likelihood estimation. The output also includes a table with standardized coefficients and the R-squared statistic.

Analyzing Imputations in R

Returning to the previous Blimp script, the `SAVE` command and the `stacked` keyword saved the imputed data sets to a single stacked file with an index variable in the first column identifying the individual files. The stacked file is appropriate for analyzing data in R, SAS, SPSS, and Stata, among others.

The R input file for the analysis is `Ex8.R`. The code block below shows the commands that import the data.

R Script Ex8.R

```
1   library(fdir)
2   library(mitml)
3   set()
4   imps <- read.table("./imps/imps.dat")
5   names(imps) <- c("imputation", "id", "condition","male","frlunch",
6     "atrisk", "stanread","efficacy", "anxiety", "mathpre", "mathpost")
```

The example requires the `fdir` and `lavaan` packages, which are loaded on lines 1 and 2. On line 3, the `set()` function of the `fdir` package identifies the file path of the folder containing the R script and sets this location as the working directory. On line 4, the `read.table` command imports the stacked data. It is only necessary to specify the name of the input data file. No file path is required when the data reside in the same folder as the R script as is the case here. Finally, variable names are listed beginning on line 5. Importantly, the first variable named `IMPUTATION` is the index that identifies the individual files.

The next block of code relies on the `mitml` package to fit the model to each data set and pool the results using Rubin's rules.

R Script Ex8.R, continued

```
 9  implist <- as.mitml.list(split(imps, imps$imputation))
10  fit <- with(implist, lm(mathpost ~ condition + frlunch
11       + efficacy + mathpre))
12  estimates <- testEstimates(fit, extra.pars = T, df.com = 245)
13  estimates
14  confint(estimates)
```

The `implist` command on line 9 unstacks the data and creates a list that contains the individual files. Line 10 fits the focal regression model using the `lm` function, and line 12 uses the `testEstimates` function in `mitml` to implement Rubin's pooling rules and save the results in an object called `estimates`. The `df.com` parameter is the denominator degrees of freedom that would have resulted had there been no missing data (i.e., $N–K–1$ degrees of freedom, where K is

the number of predictors). This argument produces Barnard and Rubin degrees of freedom values. Finally, lines 13 and 14 print the estimates and confidence intervals.

When fitting regression models to complete data sets, researchers often use an omnibus *F* test to evaluate the set of slope coefficients. The `testModels` command below specifies a multiple imputation Wald *F* statistic evaluating the null hypothesis that the population slopes equal 0 (Li et al., 1991). The test requires an additional model on line 15 that represents the null hypothesis, which in this case is an empty regression model with just an intercept. On line 16, the full model and null model objects passed into the `testModels` function, and the `D1` keyword requests the Wald test. As before, the `df.com` parameter is the denominator degrees of freedom that would have resulted had there been no missing data. This argument produces the Barnard and Rubin (1999) degrees of freedom adjustment.

R Script Ex8.R, continued

```
15   null <- with(implist, lm(mathpost ~ 1))
16   testModels(fit, null, df.com = 245, method = "D1")
```

R Output

The table of unstandardized pooled parameter estimates is shown below. The first two columns display the pooled unstandardized estimates and standard errors, and the third through fifth columns display the corresponding test statistics. The focal model results are shown in bold typeface. The `RIV` column (relative increase in variance) is a fraction comparing imputation noise to complete-data sampling variation, and the `FMI` column (fraction of missing information in the literature) quantifies the imputation noise in each estimate as proportion of its squared standard error.

```
> estimates

Call:

testEstimates(model = fit, extra.pars = T, df.com = 245)

Final parameter estimates and inferences obtained from 20 imputed data sets.

             Estimate Std.Error   t.value        df   P(>|t|)       RIV       FMI
(Intercept)    28.302     3.183     8.892   125.465     0.000     0.290     0.237
condition       2.206     1.055     2.091   133.634     0.038     0.263     0.220
frlunch        -5.392     1.094    -4.928   110.541     0.000     0.348     0.271
efficacy        0.832     0.356     2.339   103.454     0.021     0.380     0.289
mathpre         0.532     0.063     8.448   128.385     0.000     0.280     0.231

                   Estimate
Residual~~Residual   53.133

Hypothesis test adjusted for small samples with df=[245]
complete-data degrees of freedom.

> confint(estimates)
                  2.5 %     97.5 %
(Intercept) 22.0026367 34.6005103
condition    0.1192603  4.2925545
frlunch     -7.5596123 -3.2237303
efficacy     0.1266288  1.5376826
mathpre      0.4071878  0.6562495
```

The results are interpreted in the same way as a complete-data regression analysis with categorical predictors. For example, consider the slope for the treatment assignment dummy code. The positive coefficient indicates that, for two students who share the same covariate profile (i.e., lunch assistance, self-efficacy, and pretest scores), the model predicts that the student in the experimental condition should score 2.21 points higher than the student in the control group. The corresponding test statistic indicates that the slope coefficient is statistically different from zero (t = 2.09, p = .04). Note that these estimates are virtually identical to those from Bayesian and maximum likelihood estimation. Finally, the Wald omnibus F statistic is shown in the output table below. The test statistic is statistically significant, thus refuting the null hypothesis that all population slopes equal zero.

```
Model comparison calculated from 20 imputed data sets.
Combination method: D1

    F.value     df1     df2   P(>F)     RIV
     33.796       4 197.183   0.000   0.332

Hypothesis test adjusted for small samples with df=[245]
complete-data degrees of freedom.
```

EXAMPLE 9: MULTIPLE REGRESSION WITH MULTICATEGORICAL PREDICTORS

This example illustrates a multiple regression analysis with an incomplete multicategorical predictor. The analysis uses the `behaviorachievement.dat` data set taken from a longitudinal study that followed 138 students from primary through middle school. The file includes three annual assessments of broad reading and math achievement beginning in the first grade, seventh grade standardized achievement test scores taken from a statewide assessment, and a final measure of broad reading and math obtained in ninth grade. The data also contain teacher ratings of behavioral symptoms and learning problems were also obtained in the first grade. The data description at the beginning of this document provides additional details. The variables for this analysis are as follows.

Name	Definition	Missing %	Scale
	Focal Variables		
RISKGRP	Emotional/behavioral disorder risk	2.2	1 = Low, 2 = Medium, 3 = High
BEHSYMP1	1st grade behavioral symptoms	3.6	Numeric
LRNPROB1	1st grade learning problems	2.2	Numeric
READ1	1st grade broad reading composite	6.5	Numeric
READ9	9th grade broad reading composite	17.4	Numeric
	Auxiliary Variables		
READ2	2nd grade broad reading composite	9.4	Numeric
STANREAD7	7th grade standardized math	19.6	Numeric

Analysis Model

The analysis model features ninth grade broad reading scores regressed on first grade reading achievement, teacher-rated learning problems and behavioral symptoms, and a three-category nominal variable indicating risk for emotional or behavioral disorders.

$$READ_9 = \beta_0 + \beta_1(READ_1) + \beta_2(LRNPROB_1) + \beta_3(BEHSYMP_1) + \beta_4(MEDRISK) + \beta_5(HIGHRISK) + \varepsilon \quad (20)$$

The `MEDRISK` and `HIGHRISK` variables are dummy code variables that contrast the medium- and high-risk groups, respectively, against the low-risk reference group. Blimp uses a probit regression formulation that envisions multicategorical variables as arising from underlying continuous latent response difference scores. The software automatically assumes that continuous predictors and the latent response variables are multivariate normal.

The missing data literature often recommends an inclusive strategy that incorporates auxiliary variables that either predict missingness or correlate with the incomplete variables (Collins et al., 2001). Following the same factored regression specification from earlier examples (e.g., Examples 2 through 6), auxiliary variables enter the model as additional outcomes that are predicted by the analysis variables and by each other. The additional regression equations are as follows.

$$\begin{aligned} READ_2 &= \gamma_{01} + \gamma_{11}(READ9GRP) + \gamma_{21}(READ_1) \\ &+ \gamma_{31}(LRNPROB_1) + \gamma_{41}(BEHSYMP_1) + \epsilon_1 \\ STANREAD_7 &= \gamma_{02} + \gamma_{12}(READ_2) + \gamma_{22}(READ9GRP) \\ &+ \gamma_{32}(READ_1) + \gamma_{42}(LRNPROB_1) + \gamma_{52}(BEHSYMP_1) + \epsilon_2 \end{aligned} \tag{21}$$

Along with the focal regression model from Equation 20, the collection of regressions can be viewed as a path model, where the focal regression is one part of a larger network (see the path diagram from Example 2). The key difference is that the path coefficients are just a tool for linking incomplete variables and do not represent a substantive theory.

Blimp Script

The code block below shows Blimp script `Ex9.1.inp`. The first five lines can be viewed as a set of commands that specify information about the data and variables. The `DATA` command specifies the name of the input text file. No file path is required when the data file is located in the same directory as the script, as it is here. Starting on line 2, the `VARIABLES` command names the data columns. The `NOMINAL` command on line 5 identifies the multicategorical nominal predictor. By default, the group with the lowest numeric code serves as the reference category (in this example, 1 = low risk), and the user can change this specification if desired. The `MISSING` command on line 6 defines a global missing value code as 999.

Blimp Script Ex9.1.imp

```
1  DATA: behaviorachievement.dat;
2  VARIABLES: id male hispanic riskgrp atrisk behsymp1 lrnprob1
3    read1 read2 read3 read9 read9grp stanread7 math1 math2
4    math3 math9 math9grp stanmath7;
5  NOMINAL: riskgrp;
6  MISSING: 999;
7  MODEL:
8  focal.model:
9  read9 ~ read1 lrnprob1 behsymp1 riskgrp;
10 auxiliary.models:
11 stanread7 read2 ~ read9 read1 lrnprob1 behsymp1 riskgrp;
12 SEED: 90291;
13 BURN: 2000;
14 ITERATIONS: 10000;
```

The `MODEL` and `TEST` blocks can be viewed as a set. Beginning on line 7, the `MODEL` command lists the regression models, with outcome variables to the left of the tilde and predictors to the right. The code uses labels (`focal.model` and `auxiliary.models`) to order output tables, such that the focal model appears first followed by the auxiliary variable models. The focal model listed on line 9 includes the multicategorical nominal variable, which Blimp represents as a pair of dummy codes. Blimp automatically configures the explanatory variable models under the assumption that the numeric predictors and latent response variables are normally distributed. Line 11 is a syntax shortcut that produces the two auxiliary variable regression models in Equation 21; in the first model, `READ2` is regressed on the focal variables, and the second model features `STANREAD7` regressed on `READ2` and the focal variables.

Finally, lines 12 through 14 can be viewed as a block of commands that specify features of the MCMC algorithm: the `SEED` command gives an integer string that initializes the random number generator, the `BURN` command specifies the number of iterations for the warm-up or burn-in period, and the `ITERATIONS` command gives the number of MCMC iterations on which the analysis summaries are based (essentially, the number of MCMC cycles following the warm-up period).

Previous examples assigned labels to slope coefficients using the @ symbol, and these labels were subsequently used in the `TEST` command to specify custom hypothesis tests. With a

multicategorical nominal predictor, it is necessary to attach labels to individual dummy codes. To do this, you list the nominal variable's name followed by a period and a numeric suffix with each category's code value. For example, line 9 in the script would be modified as follows

```
9    read9 ~ read1@b1 lrnprob1@b2 behsymp1@b3 riskgrp.2@b4 riskgrp.3@b5;
```

where `RISKGRP.2` and `RISKGRP.3` reference the two dummy variables for the groups coded 2 and 3 in the data. The `TEST` command would then be constructed following earlier examples.

Blimp Output

Prior to inspecting the parameter estimates, it is important to investigate the potential scale reduction (PSR) factor diagnostics (Gelman & Rubin, 1992) to determine whether MCMC has converged. Blimp divides the burn-in period into 20 equal segments, and it computes the PSR diagnostic for every parameter. The table located near the top of the output reports the highest (worst) PSR value across all parameters in every model. A common recommendation is that these values should be less than 1.05 or perhaps 1.10 (Asparouhov & Muthén, 2010a; Gelman et al., 2014). If the PSR in the bottom row of the table (the final check of the burn-in period) is above these cutoffs, then rerun the analysis with a longer burn-in period.

```
BURN-IN POTENTIAL SCALE REDUCTION (PSR) OUTPUT:

  NOTE: Split chain PSR is being used. This splits each chain's
        iterations to create twice as many chains.

  Comparing iterations across 2 chains     Highest PSR   Parameter #
                            51 to 100            1.305            77
                           101 to 200            1.200            62
                           151 to 300            1.064            56
                                  ...              ...            ..
                           901 to 1800           1.017            53
                           951 to 1900           1.014            53
                          1001 to 2000           1.017            56
```

The next section of the output displays information about the variables in the analysis and the models used for estimation. This output table mimics the one from Example 6. In the interest of space, we point readers to that example for additional details. Earlier examples also show how to implement the Bayesian Wald significance test.

The table summarizing the focal regression model is shown below. The table includes unstandardized coefficients, standardized slopes, and variance explained effect size estimates.

```
OUTCOME MODEL ESTIMATES:

  Summaries based on 10000 iterations using 2 chains.

  focal.model block:

Outcome Variable:  read9

Parameters                        Median     StdDev       2.5%      97.5%        PSR      N_Eff
                                 ------------------------------------------------------------------
Variances:
  Residual Var.                   91.703     13.152     70.497    121.927      1.000   5589.273

Coefficients:
  Intercept                       68.621      6.614     55.480     81.686      1.000   5703.283
  read1                            0.484      0.049      0.389      0.581      1.000   7086.883
  lrnprob1                        -0.250      0.121     -0.485     -0.007      1.001   5583.683
  behsymp1                        -0.170      0.107     -0.379      0.042      1.000   6010.276
  riskgrp.2                       -1.682      1.991     -5.631      2.221      1.000   7073.237
  riskgrp.3                       -2.814      2.707     -8.233      2.513      1.000   6138.228

Standardized Coefficients:
  read1                            0.658      0.052      0.544      0.751      1.000   6469.725
  lrnprob1                        -0.178      0.085     -0.340     -0.005      1.001   5544.858
  behsymp1                        -0.137      0.085     -0.300      0.033      1.000   5901.722
  riskgrp.2                       -0.055      0.065     -0.182      0.073      1.000   7086.836
  riskgrp.3                       -0.079      0.075     -0.225      0.072      1.000   6182.638

Proportion Variance Explained
  by Coefficients                  0.599      0.050      0.488      0.684      1.000   5849.961
  by Residual Variation            0.401      0.050      0.316      0.512      1.000   5849.961

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

MCMC estimation produces a distribution for each parameter in the table. The median and standard deviation columns describe the center and spread of the posterior distributions; although they make no reference to drawing repeated samples, they are analogous—and numerically equivalent in most cases—to frequentist point estimates and standard errors. The

95% credible intervals in the rightmost columns give a range that captures 95% of the parameter's distribution. These are akin to confidence intervals, but the intervals describe parameter distributions rather than characteristics of repeated samples. The `N_Eff` values in rightmost column of the table give the effective number of MCMC samples for each parameter. These quantities essentially represent the number of independent estimates on which the parameter summaries are based after removing autocorrelations from the MCMC process. Gelman et al. (2014, p. 287) recommend values greater than 100. All values in the example table exceed this recommended minimum. In cases where the `N_Eff` values are insufficient, increasing the value on the `ITERATIONS` command will remedy the issue.

The results are interpreted in the same way as a complete-data regression analysis. For example, consider the first-grade reading score slope. The model predicts that two individuals who differ by one point on `READ1` but are the same on all other predictors should differ by 0.48 points on `READ9`. The 95% credible interval limits suggest this effect is statistically different from zero ($p < .05$) because the null value is well outside the interval. The two dummy codes appear as `RISKGRP.2` and `RISKGRP.3`, where the numeric suffices correspond to the numeric codes from the data. Consistent with a complete-data regression analysis, the dummy code slopes represent mean differences relative to the low-risk reference group. For example, holding all other predictors constant, the model predicts that a high-risk study would score 2.81 points lower than a low-risk student in the comparison group.

The Blimp output also includes tables of regression model parameters for the auxiliary variables as well as the auto-generated models for incomplete predictors. These additionally results are not of substantive interest and would not be reported. The auxiliary variable models appear in `OUTCOME MODEL ESTIMATES` section with the focal results, and the auto-generated predictor models are displayed under the heading `PREDICTOR MODEL ESTIMATES`.

Saving Multiple Imputations

MCMC estimation imputes missing values at every iteration, such that the resulting Bayesian estimates average over thousands of plausible replacement scores (10,000 sets in this example). A subset of the imputations can be saved for reanalysis in the frequentist framework, if desired. The Blimp input file `Ex9.2.imp` is identical `Ex9.1.imp`, but it adds the following lines.

```
NIMPS: 20;
CHAINS: 20;
SAVE:
stacked = ./imps/imps.dat;
separate = ./imps/imp*.dat;
```

The `NIMPS`, `CHAINS` and `SAVE` commands can be viewed as a set. Setting `NIMPS` equal to `CHAINS` saves a single filled-in data set from the final iteration of a unique `MCMC` process, thus avoiding autocorrelation among the imputations. The `SAVE` command provides a name for the imputed data sets. The script illustrates how to save data sets in two common formats. The stacked keyword creates a stacked file where all imputations are in a single file, and the separate keyword saves each imputed data set to a separate file with the asterisk replaced by a numeric index. To keep things organized, the `./imps` part of the file path points to a subfolder named `imps` located within the same folder as the script and data. The separate keyword also creates a list of file names needed for analysis in Mplus (in this example, a filed called `implist.dat` located in the `imps` folder).

When saving imputations, the bottom of the Blimp output file displays a table listing the order of the variables in the output data sets. All variables are saved regardless of whether they appeared in the fitted models. When saving data to a stacked file (e.g., for analysis in R or other packages), the first variable in the file is an integer index that identifies which data set each row belongs to (e.g., an integer variable that ranges from 1 to 20 in this example).

```
VARIABLE ORDER IN IMPUTED DATA:

separate = './imps/imp*.dat'

   id male hispanic riskgrp atrisk behsymp1 lrnprob1 read1 read2 read3
  read9 read9grp stanread7 math1 math2 math3 math9 math9grp stanmath7

stacked = './imps/imps.dat'

   imp# id male hispanic riskgrp atrisk behsymp1 lrnprob1 read1 read2 read3
   read9 read9grp stanread7 math1 math2 math3 math9 math9grp stanmath7
```

The imputed data sets are subsequently analyzed in another software package, and estimates and standard errors are combined using Rubin's rules (Little & Rubin, 2020). The analysis phase does not utilize the auxiliary variables, as their information is embedded in the imputations. Scripts for analyzing the imputed data sets are found in the next subsections.

Analyzing Imputations in Mplus

In lieu of the Bayesian estimates, Blimp's `SAVE` command can be used to save multiple imputations for analysis in the frequentist framework. Returning to the previous Blimp script, the `SAVE` command and the `separate` keyword saved each imputed data set to a separate file with the asterisk replaced by a numeric index. The separate keyword also creates a list of file names needed for analysis in Mplus (in this example, a filed called `implist.dat` located in the `imps` subfolder). Example 6 shows the contents of this file.

The Mplus input file for analyzing the imputations is `Ex9.inp`. The script is similar to previous Mplus scripts (e.g., the `Ex1.1.inp` file described in Example 1) with four exceptions. First, instead of naming the raw data set, the `DATA` command lists the text file containing the names of the imputed data sets (the `implist.dat` file located in the `./imps` subdirectory). The `type = imputation` subcommand instructs Mplus that the input data is a list of file names. Second, the missing subcommand is omitted because the analysis variables are now complete. Third, the `MODEL` section no longer specifies a normal distribution for the predictors or models for the auxiliary variables. Finally, lines 9 through 13 use the `DEFINE` command to create a pair of dummy codes. Lines 10 and 11 initialize a pair of new variables (`RISKGRP2` and `RISKGRP3`) with all 0s, and lines 12 and 13 recode these variables into dummy variables. Importantly, new variables computed with the `DEFINE` command must appear at the end of the `usevariables` list on line 8. The code block below shows the analysis and pooling script.

Mplus Script Ex9.inp

```
DATA:
file = ./imps/implist.dat;
type = imputation;
VARIABLE:
names = id male hispanic riskgrp atrisk behsymp1 lrnprob1
  read1 read2 read3 read9 read9grp stanread7
  math1 math2 math3 math9 math9grp stanmath7;
usevariables = read9 read1 lrnprob1 behsymp1 riskgrp2 riskgrp3;
DEFINE:
riskgrp2 = 0;
riskgrp3 = 0;
if(riskgrp eq 2) then riskgrp2 = 1;
if(riskgrp eq 3) then riskgrp3 = 1;
MODEL:
read9 on read1 lrnprob1 behsymp1 riskgrp2 riskgrp3 (beta1-beta5);
MODEL TEST:
0 = beta1; 0 = beta2; 0 = beta3;
OUTPUT:
stdyx cinterval;
```

Mplus Output

When fitting regression models to complete data sets, researchers often use an omnibus *F* test to evaluate the set of slope coefficients. The `MODEL TEST` command specified a multiple imputation Wald chi-square statistic evaluating the null hypothesis that the population slopes equal 0 (Asparouhov & Muthén, 2010b). The chi-square statistic, degrees of freedom, and *p*-value appear near the bottom of the `MODEL FIT INFORMATION` section under the `Wald Test of Parameter Constraints` heading. The test statistic is statistically significant, thus refuting the null hypothesis.

```
MODEL FIT INFORMATION

Number of Free Parameters                        7

...

Wald Test of Parameter Constraints

          Value                            173.432
          Degrees of Freedom                     5
          P-Value                           0.0000
```

The table of unstandardized parameter estimates is shown below. The first two columns display the pooled unstandardized estimates and standard errors, and the third and fourth columns display the corresponding *z*-statistics and *p*-values. The focal model results are shown in bold typeface. The `Rate of Missing` column (also called the fraction of missing information in the literature) quantifies the imputation noise in each estimate as proportion of its squared standard error.

```
MODEL RESULTS

                                                    Two-Tailed   Rate of
                  Estimate       S.E.  Est./S.E.    P-Value    Missing

 READ9    ON
    READ1            0.477      0.047     10.122      0.000      0.134
    LRNPROB1        -0.250      0.115     -2.173      0.030      0.172
    BEHSYMP1        -0.166      0.106     -1.566      0.117      0.228
    RISKGRP2        -1.710      1.882     -0.908      0.364      0.076
    RISKGRP3        -3.115      2.820     -1.105      0.269      0.272

 Intercepts
    READ9           69.174      6.218     11.125      0.000      0.154

 Residual Variances
    READ9           85.516     11.867      7.206      0.000      0.249
```

The results are interpreted in the same way as a complete-data regression analysis. For example, consider the first-grade reading score slope. The model predicts that two individuals who differ by one point on `READ1` but are the same on all other predictors should differ by 0.48 points on `READ9`. The corresponding test statistic indicates that the slope coefficient is statistically different from zero ($z = 10.29$, $p < .001$). The two dummy codes appear as `RISKGRP2` and `RISKGRP3`. Consistent with a complete-data regression analysis, the dummy code slopes represent mean differences relative to the low-risk reference group. For example, holding all other predictors constant, the model predicts that a high-risk study would score 3.12 points lower than a low-risk student in the comparison group. Note that these estimates are virtually identical to those from Bayesian estimation. The output also includes a table with standardized coefficients and the *R*-squared statistic.

Analyzing Imputations in R

Returning to the previous Blimp script, the `SAVE` command and the `stacked` keyword saved the imputed data sets to a single stacked file with an index variable in the first column identifying the individual files. The stacked file is appropriate for analyzing data in R, SAS, SPSS, and Stata, among others.

The R input file for the analysis is `Ex9.R`. The code block below shows the commands that import the data.

R Script Ex9.R

```
library(fdir)
library(mitml)
set()
imps <- read.table("./imps/imps.dat")
names(imps) <- c("imputation","id","male","hispanic","riskgrp",
  "atrisk","behsymp1","lrnprob1","read1","read2","read3",
  "read9","read9grp","stanread7","math1","math2","math3",
  "math9","math9grp","stanmath7")
imps$riskgrp <- factor(imps$riskgrp)
```

The example requires the `fdir` and `lavaan` packages, which are loaded on lines 1 and 2. On line 3, the `set()` function of the `fdir` package identifies the file path of the folder containing the R script and sets this location as the working directory. On line 4, the `read.table` command imports the stacked data. It is only necessary to specify the name of the input data file. No file path is required when the data reside in the same folder as the R script as is the case here. Variable names are listed beginning on line 5. Importantly, the first variable named `IMPUTATION` is the index that identifies the individual files. Finally, line 9 defines the `RISKGRP` variable as a factor with qualitatively different levels. This specification will automatically introduce a set of dummy codes into the regression model.

The next block of code relies on the `mitml` package to fit the model to each data set and pool the results using Rubin's rules.

R Script Ex9.R, continued

```
10  implist <- as.mitml.list(split(imps, imps$imputation))
11  fit <- with(implist, lm(read9 ~ read1 + lrnprob1 + behsymp1 + riskgrp))
12  estimates <- testEstimates(fit, extra.pars = T, df.com = 132)
13  estimates
14  confint(estimates)
```

The `implist` command on line 10 unstacks the data and creates a list that contains the individual files. Line 11 fits the focal regression model using the `lm` function, and line 12 uses the `testEstimates` function in `mitml` to implement Rubin's pooling rules and save the results in an object called `estimates`. The `df.com` parameter is the denominator degrees of freedom that would have resulted had there been no missing data (i.e., $N–K–1$ degrees of freedom, where K is the number of predictors). This argument produces Barnard and Rubin degrees of freedom values. Finally, lines 13 and 14 print the estimates and confidence intervals.

When fitting regression models to complete data sets, researchers often use an omnibus F test to evaluate the set of slope coefficients. The `testModels` command below specifies a multiple imputation Wald F statistic evaluating the null hypothesis that the population slopes equal 0 (Li et al., 1991). The test requires an additional model on line 15 that represents the null hypothesis, which in this case is an empty regression model with just an intercept. On line 16, the full model and null model objects passed into the `testModels` function, and the `D1` keyword requests the Wald test. As before, the `df.com` parameter is the denominator degrees of freedom that would

have resulted had there been no missing data. This argument produces the Barnard and Rubin (1999) degrees of freedom adjustment.

R Script Ex9.R, continued

```
15   null <- with(implist, lm(read9 ~ 1))
16   testModels(fit, null, df.com = 132, method = "D1")
```

R Output

The table of unstandardized pooled parameter estimates is shown below. The first two columns display the pooled unstandardized estimates and standard errors, and the third through fifth columns display the corresponding test statistics. The focal model results are shown in bold typeface. The `RIV` column (relative increase in variance) is a fraction comparing imputation noise to complete-data sampling variation, and the `FMI` column (fraction of missing information in the literature) quantifies the imputation noise in each estimate as proportion of its squared standard error.

```
> estimates

Call:

testEstimates(model = fit, extra.pars = T, df.com = 132)

Final parameter estimates and inferences obtained from 20 imputed data sets.

             Estimate Std.Error   t.value        df   P(>|t|)       RIV       FMI
(Intercept)    69.174     6.337    10.916    98.577     0.000     0.172     0.164
read1           0.477     0.048     9.928   103.392     0.000     0.146     0.144
lrnprob1       -0.250     0.117    -2.133    94.276     0.036     0.196     0.181
behsymp1       -0.166     0.108    -1.539    81.473     0.128     0.276     0.235
riskgrp2       -1.710     1.921    -0.890   116.647     0.375     0.079     0.088
riskgrp3       -3.115     2.867    -1.087    72.059     0.281     0.348     0.278

                   Estimate
Residual~~Residual   89.403

Hypothesis test adjusted for small samples with df=[132]
complete-data degrees of freedom.
```

```
> confint(estimates)
                 2.5 %      97.5 %
(Intercept) 56.5999199 81.74779632
read1        0.3820806  0.57283035
lrnprob1    -0.4822008 -0.01730107
behsymp1    -0.3796984  0.04849960
riskgrp2    -5.5147538  2.09562507
riskgrp3    -8.8300720  2.59967016
```

The results are interpreted in the same way as a complete-data regression analysis. For example, consider the first-grade reading score slope. The model predicts that two individuals who differ by one point on `READ1` but are the same on all other predictors should differ by 0.48 points on `READ9`. The corresponding test statistic indicates that the slope coefficient is statistically different from zero ($t = 9.93$, $p < .001$). The two dummy codes appear as `RISKGRP2` and `RISKGRP3`. Consistent with a complete-data regression analysis, the dummy code slopes represent mean differences relative to the low-risk reference group. For example, holding all other predictors constant, the model predicts that a high-risk study would score 3.12 points lower than a low-risk student in the comparison group. Note that these estimates are virtually identical to those from Bayesian and maximum likelihood estimation.

Finally, the Wald omnibus *F* statistic is shown in the output table below. The test statistic is statistically significant, thus refuting the null hypothesis that all population slopes equal zero.

```
Model comparison calculated from 20 imputed data sets.
Combination method: D1

    F.value     df1     df2   P(>F)     RIV
     33.252       5 123.203   0.000   0.213

Hypothesis test adjusted for small samples with df=[132]
complete-data degrees of freedom.
```

EXAMPLE 10: MULTIPLE REGRESSION WITH AN INTERACTION EFFECT

This example illustrates a multiple regression analysis with an incomplete interaction effect. The analysis uses the `behaviorachievement.dat` data set taken from a longitudinal study that followed 138 students from primary through middle school. The file includes three annual assessments of broad reading and math achievement beginning in the first grade, seventh grade standardized achievement test scores taken from a statewide assessment, and a final measure of broad reading and math obtained in ninth grade. The data also contain teacher ratings of behavioral symptoms and learning problems were also obtained in the first grade. The data description at the beginning of this document provides additional details. The variables for this analysis are as follows.

Name	Definition	Missing %	Scale
	Focal Variables		
BEHSYMP1	1st grade behavioral symptoms	3.6	Numeric
LRNPROB1	1st grade learning problems	2.2	Numeric
READ1	1st grade broad reading composite	6.5	Numeric
READ9	9th grade broad reading composite	17.4	Numeric
	Auxiliary Variables		
READ2	2nd grade broad reading composite	9.4	Numeric
STANREAD7	7th grade standardized math	19.6	Numeric

Analysis Model

The analysis model features ninth grade broad reading scores regressed on first grade reading achievement, teacher-rated learning problems and behavioral symptoms, and the product of first grade reading scores and learning problems.

$$\begin{aligned} READ_9 &= \beta_0 + \beta_1(READ_1) + \beta_2(LRNPROB_1) \\ &+ \beta_3(READ_1)(LRNPROB_1) + \beta_4(BEHSYMP_1) + \varepsilon \end{aligned} \tag{22}$$

Unlike a complete-data regression analysis, all incomplete variables require distributional assumptions, including the predictors. Moderated regression models (and models with non-

linearities more generally) require a factored regression specification that assigns separate distributions to the predictors and outcome. By default, Blimp invokes a multivariate normal distribution for incomplete predictors. Importantly, the product term does not require a unique distribution, as missing data imputation generates lower-order variables that preserve the interaction effect in the focal model.

The missing data literature often recommends an inclusive strategy that incorporates auxiliary variables that either predict missingness or correlate with the incomplete variables (Collins et al., 2001). Following earlier examples, auxiliary variables enter the model as additional outcomes that are predicted by the analysis variables and by each other. The additional regression equations are as follows.

$$\begin{aligned} READ_2 &= \gamma_{01} + \gamma_{11}(READ9GRP) + \gamma_{21}(READ_1) \\ &+ \gamma_{31}(LRNPROB_1) + \gamma_{41}(BEHSYMP_1) + \epsilon_1 \\ STANREAD_7 &= \gamma_{02} + \gamma_{12}(READ_2) + \gamma_{22}(READ9GRP) \\ &+ \gamma_{32}(READ_1) + \gamma_{42}(LRNPROB_1) + \gamma_{52}(BEHSYMP_1) + \epsilon_2 \end{aligned} \tag{23}$$

Along with the other models, the collection of regression equations can be viewed as a path model where the focal analysis is one part of a larger network (see the path diagram from Example 2). The key difference is that the path coefficients are just a tool for linking incomplete variables and do not represent a substantive theory.

Blimp Script

The code block below shows Blimp script `Ex10.1.inp`. The first five lines can be viewed as a set of commands that specify information about the data and variables. The `DATA` command specifies the name of the input text file. No file path is required when the data file is located in the same directory as the script, as it is here. Starting on line 2, the `VARIABLES` command names the data columns, and `MISSING` command on line 5 defines a global missing value code as 999.

Blimp Script Ex10.1.imp

```
1  DATA: behaviorachievement.dat;
2  VARIABLES: id male hispanic riskgrp atrisk behsymp1 lrnprob1
3    read1 read2 read3 read9 read9grp stanread7
4    math1 math2 math3 math9 math9grp stanmath7;
5  MISSING: 999;
6  CENTER: read1 lrnprob1;
7  MODEL:
8  focal.model:
9  read9 ~ read1 lrnprob1 read1*lrnprob1@beta3 behsymp1;
10 auxiliary.model:
11 stanread7 read2 ~ read9 read1 lrnprob1 behsymp1;
12 SIMPLE: read1 | lrnprob1;
13 TEST:
14 beta3 = 0;
15 SEED: 90291;
16 BURN: 5000;
17 ITERATIONS: 10000;
```

The `CENTER`, `MODEL`, `SIMPLE`, and `TEST` blocks can be viewed as a set. The `CENTER` command deviates the two interacting variables at their iteratively-estimated grand means. Beginning on line 7, the `MODEL` command lists the regression models, with outcome variables to the left of the tilde and predictors to the right. The code uses labels (`focal.model` and `auxiliary.models`) to order output tables, such that the focal model appears first followed by the auxiliary variable models. The focal model listed on line 9 includes a product term, which is specified by joining two variables with an asterisk. The product's slope coefficient is labeled using the @ symbol. Blimp automatically configures the explanatory variable models under the assumption that they are normally distributed. Line 11 is a syntax shortcut that produces the two auxiliary variable regression models in Equation 23; in the first model, `READ2` is regressed on the focal variables, and the second model features `STANREAD7` regressed on `READ2` and the focal variables. The `SIMPLE` command requests the conditional effects (i.e., simple slopes) of `READ1` at different levels of `LRNPROB1`. By default, Blimp adopts a pick-a-point approach that uses standard deviation units of the moderator variable, although the user can specify custom values. Finally, the `TEST` command uses the parameter label to specify a custom hypothesis test that the interaction slope equals zero. This command produces the Bayesian Wald test (Asparouhov & Muthén, 2021),

which is essentially a chi-square statistic that captures the discrepancy between the Bayesian point estimate (posterior means) and the hypothesized value of zero.

Finally, lines 15 through 17 can be viewed as a block of commands that specify features of the MCMC algorithm: the `SEED` command gives an integer string that initializes the random number generator, the `BURN` command specifies the number of iterations for the warm-up or burn-in period, and the `ITERATIONS` command gives the number of MCMC iterations on which the analysis summaries are based (essentially, the number of MCMC cycles following the warm-up period).

Blimp Output

Prior to inspecting the parameter estimates, it is important to investigate the potential scale reduction (PSR) factor diagnostics (Gelman & Rubin, 1992) to determine whether MCMC has converged. Blimp divides the burn-in period into 20 equal segments, and it computes the PSR diagnostic for every parameter. The table located near the top of the output reports the highest (worst) PSR value across all parameters in every model. A common recommendation is that these values should be less than 1.05 or perhaps 1.10 (Asparouhov & Muthén, 2010a; Gelman et al., 2014). If the PSR in the bottom row of the table (the final check of the burn-in period) is above these cutoffs, then rerun the analysis with a longer burn-in period.

```
BURN-IN POTENTIAL SCALE REDUCTION (PSR) OUTPUT:

  NOTE: Split chain PSR is being used. This splits each chain's
        iterations to create twice as many chains.

  Comparing iterations across 2 chains     Highest PSR   Parameter #
                           126 to 250           1.291            47
                           251 to 500           1.156            47
                           376 to 750           1.053            43
                           501 to 1000          1.042            43
                                   ...            ...            ..
                          2251 to 4500          1.015            43
                          2376 to 4750          1.009            47
                          2501 to 5000          1.016            47
```

The next section of the output displays information about the variables in the analysis and the models used for estimation. This output table mimics the one from Example 6. In the interest of space, we point readers to that example for additional details.

Thus far, the examples have strictly used 95% credible intervals for evaluating the significance of individual slope coefficients. The `TEST` command in the previous script requested a Bayesian Wald chi-square statistic (Asparouhov & Muthén, 2021) that evaluates the null hypothesis that the population interaction effect equals zero. The chi-square statistic, degrees of freedom, and *p*-value appear near the bottom of the `MODEL FIT` section under the `WALD TEST` heading. The test statistic is statistically significant, thus refuting the null hypothesis.

```
MODEL FIT:

  ...

  WALD TESTS (Asparouhov & Muthén, 2021)

  Test #1

    Full:
      [1]  read9 ~ Intercept read1 lrnprob1 behsymp1 read1*lrnprob1@beta3

    Restricted:
      [1]  read9 ~ Intercept read1 lrnprob1 behsymp1 read1*lrnprob1@beta3

    Constraints in Restricted:
      [1]  beta3 = 1

    Wald Statistic (Chi-Square)                 7.050
    Number of Parameters Tested (df)                1
    Probability                                 0.008
```

The table summarizing the focal regression model is shown below. The table includes unstandardized coefficients, standardized slopes, and variance explained effect size estimates.

```
OUTCOME MODEL ESTIMATES:

  Summaries based on 10000 iterations using 2 chains.

  focal.model block:

Outcome Variable:  read9

Grand Mean Centered: lrnprob1 read1


Parameters                          Median     StdDev       2.5%      97.5%        PSR      N_Eff
                                  ------------------------------------------------------------------
Variances:
  Residual Var.                     84.845     12.207     65.395    113.644      1.000   5244.287

Coefficients:
  Intercept                         95.043      5.232     84.708    105.341      1.000   3721.278
  read1                              0.514      0.045      0.427      0.603      1.002   3166.488
  lrnprob1                          -0.281      0.119     -0.513     -0.045      1.000   3866.502
  behsymp1                          -0.146      0.103     -0.346      0.056      1.000   5731.723
  read1*lrnprob1                     0.012      0.005      0.003      0.021      1.000   3214.340

Standardized Coefficients:
  read1                              0.687      0.042      0.596      0.760      1.002   2549.527
  lrnprob1                          -0.200      0.083     -0.361     -0.032      1.000   3886.718
  behsymp1                          -0.117      0.081     -0.275      0.044      1.000   5711.694
  read1*lrnprob1                     0.167      0.059      0.049      0.280      1.000   4120.699

Proportion Variance Explained
  by Coefficients                    0.630      0.048      0.525      0.711      1.001   4765.860
  by Residual Variation              0.370      0.048      0.289      0.475      1.001   4765.860

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

MCMC estimation produces a distribution for each parameter in the table. The median and standard deviation columns describe the center and spread of the posterior distributions; although they make no reference to drawing repeated samples, they are analogous—and numerically equivalent in most cases—to frequentist point estimates and standard errors. The 95% credible intervals in the rightmost columns give a range that captures 95% of the parameter's distribution. These are akin to confidence intervals, but the intervals describe parameter distributions rather than characteristics of repeated samples. The `N_Eff` values in rightmost column of the table give the effective number of MCMC samples for each parameter. These quantities essentially represent the number of independent estimates on which the parameter summaries are based after removing autocorrelations from the MCMC process. Gelman et al. (2014, p. 287) recommend values greater than 100. All values in the example table

exceed this recommended minimum. In cases where the `N_Eff` values are insufficient, increasing the value on the `ITERATIONS` command will remedy the issue.

The lower-order terms in a moderated regression are conditional effects that depend on scaling or centering. Specifically, the lower-order slope of first grade reading scores ($\beta_1 = 0.51$) is the effect of that predictor at the mean of the first-grade learning problems, and the learning problems slope ($\beta_2 = -0.28$) similarly reflects the conditional effect at the reading score mean. The interaction slope captures the change in the first-grade reading slope for each one-unit increase in learning problems (and vice versa). Specifically, the positive coefficient ($\beta_3 = 0.012$) indicates that the association between first and ninth grade reading scores becomes stronger (i.e., more positive) as learning problems increase. That is, the predictive power of early reading on later reading is strongest for students with elevated learning problem ratings in first grade. Consistent with the Wald test statistic, the 95% credible interval limits suggest this effect is statistically different from zero ($p < .05$) because the null value is well outside the interval.

The `SIMPLE` command prints a table of conditional effects (simple slopes) of `READ1` at different standard deviation units of `LRNPROB1`. The output is shown below.

```
Conditional Effects                Median     StdDev       2.5%      97.5%        PSR      N_Eff
                                  ------------------------------------------------------------------
  read1 | lrnprob1 @ +2 SD
    Intercept                      88.906      7.322     74.527    103.228      1.000   3564.186
    Slope                           0.775      0.115      0.560      1.012      1.001   2734.321

  read1 | lrnprob1 @ +1 SD
    Intercept                      91.999      6.229     79.633    104.181      1.000   3595.490
    Slope                           0.645      0.071      0.510      0.791      1.002   2562.360

  read1 | lrnprob1 @ 0
    Intercept                      95.043      5.232     84.708    105.341      1.000   3721.278
    Slope                           0.514      0.045      0.427      0.603      1.002   3166.488

  read1 | lrnprob1 @ -1 SD
    Intercept                      98.097      4.397     89.418    106.685      1.000   4067.215
    Slope                           0.383      0.063      0.257      0.505      1.001   4977.884

  read1 | lrnprob1 @ -2 SD
    Intercept                     101.157      3.831     93.636    108.667      1.000   4939.955
    Slope                           0.252      0.105      0.036      0.453      1.000   4319.446

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

Consistent with the positive interaction coefficient, the simple slopes increase in strength as learning problems ratings increase (and vice versa). All of the tabled conditional effects are statistically significant at $p < .05$ because the null value does not fall within the 95% credible intervals.

The Blimp output also includes tables of regression model parameters for the auxiliary variables as well as the auto-generated models for incomplete predictors. These additionally results are not of substantive interest and would not be reported. The auxiliary variable models appear in `OUTCOME MODEL ESTIMATES` section with the focal results, and the auto-generated predictor models are displayed under the heading `PREDICTOR MODEL ESTIMATES`.

Saving Multiple Imputations

MCMC estimation imputes missing values at every iteration, such that the resulting Bayesian estimates average over thousands of plausible replacement scores (10,000 sets in this example). A subset of the imputations can be saved for reanalysis in the frequentist framework, if desired. The Blimp input file `Ex10.2.imp` is identical `Ex10.1.imp`, but it adds the following lines at the bottom of the script.

```
NIMPS: 20;
CHAINS: 20;
SAVE:
stacked = ./imps/imps.dat;
separate = ./imps/imp*.dat;
```

The `NIMPS`, `CHAINS` and `SAVE` commands can be viewed as a set. Setting `NIMPS` equal to `CHAINS` saves a single filled-in data set from the final iteration of a unique `MCMC` process, thus avoiding autocorrelation among the imputations. The `SAVE` command provides a name for the imputed data sets. The script illustrates how to save data sets in two common formats. The stacked keyword creates a stacked file where all imputations are in a single file, and the separate keyword saves each imputed data set to a separate file with the asterisk replaced by a numeric index. To keep things organized, the `./imps` part of the file path points to a subfolder named `imps` located within the same folder as the script and data. The separate keyword also creates a list of file names needed for analysis in Mplus (in this example, a filed called `implist.dat` located in the `imps` folder).

When saving imputations, the bottom of the Blimp output file displays a table listing the order of the variables in the output data sets. All variables are saved regardless of whether they appeared in the fitted models. When saving data to a stacked file (e.g., for analysis in R or other packages), the first variable in the file is an integer index that identifies which data set each row belongs to (e.g., an integer variable that ranges from 1 to 20 in this example).

```
VARIABLE ORDER IN IMPUTED DATA:

separate = './imps/imp*.dat'

   id male hispanic riskgrp atrisk behsymp1 lrnprob1 read1 read2 read3
  read9 read9grp stanread7 math1 math2 math3 math9 math9grp stanmath7

stacked = './imps/imps.dat'

   imp# id male hispanic riskgrp atrisk behsymp1 lrnprob1 read1 read2 read3
   read9 read9grp stanread7 math1 math2 math3 math9 math9grp stanmath7
```

The imputed data sets are subsequently analyzed in another software package, and estimates and standard errors are combined using Rubin's rules (Little & Rubin, 2020). The analysis phase does not utilize the auxiliary variables, as their information is embedded in the imputations. Scripts for analyzing the imputed data sets are found in the next subsections.

Analyzing Imputations in Mplus

In lieu of the Bayesian estimates, Blimp's `SAVE` command can be used to save multiple imputations for analysis in the frequentist framework. Returning to the previous Blimp script, the `SAVE` command and the `separate` keyword saved each imputed data set to a separate file with the asterisk replaced by a numeric index. The separate keyword also creates a list of file names needed for analysis in Mplus (in this example, a filed called `implist.dat` located in the `imps` subfolder). Example 6 shows the contents of this file.

The Mplus input file for analyzing the imputations is `Ex10.inp`. The major commands are similar to the `Ex6.1.inp` file described in Example 6. Consistent with previous multiple imputation analysis scripts, the `DATA` command lists the text file containing the names of the imputed data sets (the `implist.dat` file located in the `./imps` subdirectory). The `type =`

`imputation` subcommand instructs Mplus that the input data is a list of file names. Second, the missing subcommand is omitted because the analysis variables are now complete. Third, the `MODEL` section no longer specifies a normal distribution for the predictors or models for the auxiliary variables. The code block below shows the analysis and pooling script.

Mplus Script Ex10.inp

```
1   DATA:
2   file = ./imps/implist.dat;
3   type = imputation;
4   VARIABLE:
5   names = id male hispanic riskgrp atrisk behsymp1 lrnprob1
6     read1 read2 read3 read9 read9grp stanread7
7     math1 math2 math3 math9 math9grp stanmath7;
8   usevariables = read9 read1 lrnprob1 behsymp1 product;
9   DEFINE:
10  center read1 lrnprob1 (grandmean);
11  product = read1 * lrnprob1;
12  MODEL:
13  read9 on read1 lrnprob1 product behsymp1 (beta1-beta4);
14  MODEL CONSTRAINT:
15  new(lrnprobvar slp_low slp_mean slp_high);
16  lrnprobvar = 114.354;
17  slp_high = beta1 + beta3*1*sqrt(lrnprobvar);
18  slp_mean = beta1 + beta3*0*sqrt(lrnprobvar);
19  slp_low =  beta1 - beta3*1*sqrt(lrnprobvar);
19  OUTPUT:
20  stdyx cinterval;
```

The script also invokes several new features. On line 10, the `center` subcommand under the `DEFINE` command centers the two interacting predictors at their grand means, and line 11 computes a new variable equal to the product of the centered scores. Importantly, new variables computed with the `DEFINE` command must appear at the end of the `usevariables` list on line 8. Beginning on line 14, the `MODEL CONSTRAINT` command is used to compute conditional effects or simple slopes. First, line 15 assigns names to four new parameters (the variance of the moderator and three simple slopes). Line 16 inputs the variance of the moderator (obtained from

the descriptive statistics on the output), and lines 17 through 20 compute the conditional effect of `READ1` at the mean of `LRNPROB1` and at plus and minus one standard deviation from the mean.

Mplus Output

The table of unstandardized parameter estimates is shown below. The first two columns display the pooled unstandardized estimates and standard errors, and the third and fourth columns display the corresponding *z*-statistics and *p*-values. The focal model results are shown in bold typeface. The `Rate of Missing` column (also called the fraction of missing information in the literature) quantifies the imputation noise in each estimate as proportion of its squared standard error.

```
MODEL RESULTS

                                                    Two-Tailed   Rate of
                  Estimate       S.E.  Est./S.E.    P-Value    Missing

 READ9    ON
    READ1            0.513      0.041     12.423      0.000      0.166
    LRNPROB1        -0.287      0.112     -2.568      0.010      0.198
    READ1BYLPR       0.012      0.004      3.035      0.002      0.168
    BEHSYMP1        -0.144      0.097     -1.479      0.139      0.161

 Intercepts
    READ9           94.885      4.913     19.314      0.000      0.172

 Residual Variances
    READ9           78.835     10.144      7.772      0.000      0.124
```

The lower-order terms in a moderated regression are conditional effects that depend on scaling or centering. Specifically, the lower-order slope of first grade reading scores ($\hat{\beta}_1 = 0.51$) is the effect of that predictor at the mean of the first-grade learning problems, and the learning problems slope ($\hat{\beta}_2 = -0.29$) similarly reflects the conditional effect at the reading score mean. The interaction slope captures the change in the first-grade reading slope for each one-unit increase in learning problems (and vice versa). Specifically, the positive coefficient ($\hat{\beta}_3 = 0.012$) indicates that the association between first and ninth grade reading scores becomes stronger (i.e., more positive) as learning problems increase. That is, the predictive power of early reading on

later reading is strongest for students with elevated learning problem ratings in first grade. Note that these estimates are numerically identical to those from Bayesian and maximum likelihood estimation. The output also includes a table with standardized coefficients and the *R*-squared statistic.

Finally, the printed output also includes the table of conditional effects, which were computed using the `MODEL CONSTRAINT` command. The output is shown below. Consistent with the positive interaction coefficient, the simple slopes increase in strength as learning problems ratings increase (and vice versa). All of the tabled conditional effects are statistically significant at $p < .05$.

```
                                                  Two-Tailed   Rate of
                    Estimate       S.E.  Est./S.E.    P-Value    Missing
...

New/Additional Parameters
    LRNPROBV         114.354      0.022   5129.998      0.000      1.000
    SLP_LOW            0.382      0.058      6.619      0.000      0.064
    SLP_MEAN           0.513      0.041     12.423      0.000      0.166
    SLP_HIGH           0.644      0.062     10.425      0.000      0.259
```

Analyzing Imputations in R

Returning to the previous Blimp script, the `SAVE` command and the `stacked` keyword saved the imputed data sets to a single stacked file with an index variable in the first column identifying the individual files. The stacked file is appropriate for analyzing data in R, SAS, SPSS, and Stata, among others.

The R input file for the analysis is `Ex10.R`. The code block below shows the commands that import the data. The example requires the `fdir` and `mitml` packages, which are loaded on lines 1 and 2. On line 3, the `set()` function of the `fdir` package identifies the file path of the folder containing the R script and sets this location as the working directory. On line 4, the `read.table` command imports the stacked data. It is only necessary to specify the name of the input data file. No file path is required when the data reside in the same folder as the R script as is the case here. Variable names are listed beginning on line 5. Importantly, the first variable named `IMPUTATION` is the index that identifies the individual files. Finally, lines 9 and 10 create new

centered variables called `READ1.CGM` and `LRNPROB1.CGM`, and line 10 computes the product of the centered variables.

R Script Ex10.R

```
1   library(fdir)
2   library(mitml)
3   set()
4   imps <- read.table("./imps/imps.dat")
5   names(imps) <- c("imputation","id","male","hispanic","riskgrp",
6     "atrisk","behsymp1","lrnprob1","read1","read2","read3",
7     "read9","read9grp","stanread7","math1","math2","math3",
8     "math9","math9grp","stanmath7")
9   imps$read1.cgm <- imps$read1 - mean(imps$read1)
10  imps$lrnprob1.cgm <- imps$lrnprob1 - mean(imps$lrnprob1)
11  imps$product <- imps$read1.cgm * imps$lrnprob1.cgm
```

The next block of code relies on the `mitml` package to fit the model to each data set and pool the results using Rubin's rules.

R Script Ex10.R, continued

```
12  implist <- as.mitml.list(split(imps, imps$imputation))
13  fit <- with(implist, lm(read9 ~ read1.cgm + lrnprob1.cgm +
14    product + behsymp1))
15  estimates <- testEstimates(fit, extra.pars = T, df.com = 133)
16  estimates
17  confint(estimates)
```

The `implist` command on line 12 unstacks the data and creates a list that contains the individual files. Lines 13 and 14 fit the focal regression model using the `lm` function, and line 15 uses the `testEstimates` function in `mitml` to implement Rubin's pooling rules and save the results in an object called `estimates`. The `df.com` parameter is the denominator degrees of freedom that would have resulted had there been no missing data (i.e., $N–K–1$ degrees of freedom, where K is the number of predictors). This argument produces Barnard and Rubin degrees of freedom values. Finally, lines 16 and 17 print the estimates and confidence intervals.

The final code block below computes conditional effects or simple slopes. Line 18 computes the pooled standard deviation of the moderator (`LRNPROB1.CGM`). Line 19 prints the value, which equals 10.77. Lines 20, 22, and 24 define text strings that define the computation of the conditional effect of `READ1` at the mean of `LRNPROB1` and at plus and minus one standard deviation from the mean. Lines 21, 23, and 25 use the `testConstraints` function in `mitml` to compute the pooled coefficients and test statistics.

R Script Ex10.R, continued

```
18  lrnprob1.sd <- mean(unlist(lapply(implist, (function(x) sd(x$lrnprob1.cgm)))))
19  lrnprob1.sd
20  slp_high <- "read1.cgm + product*1*10.77"
21  testConstraints(fit, constraints = slp_high, df.com = 133)
22  slp_mean <- "read1.cgm + product*0*10.77"
23  testConstraints(fit, constraints = slp_mean, df.com = 133)
24  slp_low <- "read1.cgm + product*-1*10.77"
25  testConstraints(fit, constraints = slp_low, df.com = 133)
```

R Output

The table of unstandardized pooled parameter estimates is shown below. The first two columns display the pooled unstandardized estimates and standard errors, and the third through fifth columns display the corresponding test statistics. The focal model results are shown in bold typeface. The `RIV` column (relative increase in variance) is a fraction comparing imputation noise to complete-data sampling variation, and the `FMI` column (fraction of missing information in the literature) quantifies the imputation noise in each estimate as proportion of its squared standard error.

```
> estimates

Call:

testEstimates(model = fit, extra.pars = T, df.com = 133)

Final parameter estimates and inferences obtained from 20 imputed data sets.

              Estimate Std.Error   t.value        df   P(>|t|)       RIV       FMI
(Intercept)     94.885     4.993    19.005    94.305     0.000     0.199     0.183
read1.cgm        0.513     0.042    12.226    95.990     0.000     0.190     0.176
lrnprob1.cgm    -0.287     0.114    -2.524    87.532     0.013     0.240     0.211
behsymp1        -0.144     0.099    -1.457    97.506     0.148     0.181     0.170
product          0.012     0.004     2.989    95.632     0.004     0.192     0.178

                   Estimate
Residual~~Residual   81.798

Hypothesis test adjusted for small samples with df=[133]
complete-data degrees of freedom.

> confint(estimates)
                    2.5 %       97.5 %
(Intercept)  84.97267773 104.79773864
read1.cgm     0.42938742   0.59584085
lrnprob1.cgm -0.51234117  -0.06092844
behsymp1     -0.33979788   0.05214293
product       0.00411505   0.02039524
```

The lower-order terms in a moderated regression are conditional effects that depend on scaling or centering. Specifically, the lower-order slope of first grade reading scores ($\hat{\beta}_1 = 0.51$) is the effect of that predictor at the mean of the first-grade learning problems, and the learning problems slope ($\hat{\beta}_2 = -0.29$) similarly reflects the conditional effect at the reading score mean. The interaction slope captures the change in the first-grade reading slope for each one-unit increase in learning problems (and vice versa). Specifically, the positive coefficient ($\hat{\beta}_3 = 0.012$) indicates that the association between first and ninth grade reading scores becomes stronger (i.e., more positive) as learning problems increase. That is, the predictive power of early reading on later reading is strongest for students with elevated learning problem ratings in first grade. Note that these estimates are numerically identical to those from Bayesian and maximum likelihood estimation. The output also includes a table with standardized coefficients and the *R*-squared statistic.

Finally, the printed output also includes the table of conditional effects. The output is shown below. Consistent with the positive interaction coefficient, the simple slopes increase in strength

as learning problems ratings increase (and vice versa). All of the tabled conditional effects are statistically significant at $p < .05$.

```
testConstraints(model = fit, constraints = slp_high, df.com = 133)

Hypothesis test calculated from 20 imputed data sets. The following
constraints were specified:

                                  Estimate Std. Error
   read1.cgm + product*1*10.77:      0.645      0.063

Combination method: D1

    F.value     df1     df2   P(>F)     RIV
    105.076       1  77.758   0.000   0.330

Hypothesis test adjusted for small samples with df=[133]
complete-data degrees of freedom.

testConstraints(model = fit, constraints = slp_mean, df.com = 133)

Hypothesis test calculated from 20 imputed data sets. The following
constraints were specified:

                                  Estimate Std. Error
   read1.cgm + product*0*10.77:      0.513      0.042

Combination method: D1

    F.value     df1     df2   P(>F)     RIV
    149.476       1 101.581   0.000   0.190

Hypothesis test adjusted for small samples with df=[133]
complete-data degrees of freedom.
```

```
testConstraints(model = fit, constraints = slp_low, df.com = 133)

Hypothesis test calculated from 20 imputed data sets. The following
constraints were specified:

                                    Estimate Std. Error
   read1.cgm + product*-1*10.77:      0.381      0.059

Combination method: D1

    F.value     df1     df2   P(>F)     RIV
     41.842       1 125.630   0.000   0.064

Hypothesis test adjusted for small samples with df=[133]
complete-data degrees of freedom.
```

EXAMPLE 11: CURVILINEAR REGRESSION

This example illustrates a multiple regression analysis with an incomplete curvilinear effect. The analysis uses the `mathachievement.dat` data set taken from an educational intervention where 250 students were assigned to an intervention and comparison condition. The file includes pretest and posttest math achievement scores, a measure of math self-efficacy, standardized reading scores taken from a statewide assessment, and several sociodemographic variables. The analysis variables are as follows.

Name	Definition	Missing %	Scale
	Focal Variables		
MATHPOST	Math achievement posttest	18.0	Numeric
ANXIETY	Math anxiety composite	8.4	Numeric
FRLUNCH	Lunch assistance code	4.4	0 = None, 1 = Free/reduced lunch
EFFICACY	Math self-efficacy rating	9.6	Ordinal (1 to 6)
MATHPRE	Math achievement pretest	0	Numeric
	Auxiliary Variables		
ATRISK	Behavioral disorder risk	5.2	0 = Low risk, 1 = At-risk
STANREAD	Standardized reading	9.2	Numeric

Analysis Model

The analysis model features math posttest scores regressed on anxiety and its square, the lunch assistance dummy code, math self-efficacy ratings, and math pretest scores.

$$\begin{aligned} MATHPOST = \beta_0 + \beta_1(ANXIETY) + \beta_2(ANXIETY^2) \\ + \beta_3(FRLUNCH) + \beta_4(EFFICACY) + \beta_5(MATHPRE) + \varepsilon \end{aligned} \tag{24}$$

Unlike a complete-data regression analysis, all incomplete variables require distributional assumptions, including the predictors. Curvilinear regression models (and models with non-linearities more generally) require a factored regression specification that assigns separate distributions to the predictors and outcome. By default, Blimp invokes a multivariate normal distribution for numeric predictors and latent response scores.

The missing data literature often recommends an inclusive strategy that incorporates auxiliary variables that either predict missingness or correlate with the incomplete variables (Collins et al., 2001). Following earlier examples, auxiliary variables enter the model as additional outcomes that are predicted by the analysis variables and by each other. The additional regression equations are as follows.

$$\begin{aligned} ATRISK^{*} &= \gamma_{01} + \gamma_{11}(MATHPOST) + \gamma_{21}(CONDITION) \\ &+ \gamma_{31}(FRLUNCH) + \gamma_{41}(EFFICACY) + \gamma_{51}(MATHPRE) + \epsilon_1 \\ STANREAD &= \gamma_{02} + \gamma_{12}(ATRISK) + \gamma_{22}(MATHPOST) + \gamma_{32}(CONDITION) \\ &+ \gamma_{42}(FRLUNCH) + \gamma_{52}(EFFICACY) + \gamma_{62}(MATHPRE) + \epsilon_2 \end{aligned} \tag{25}$$

The `ATRISK` model is a probit regression, with the binary outcome model as a latent response variable (denoted by the asterisk superscript). Again, the entire collection of regressions can be viewed as a path model, where the focal regression is one part of a larger network (see the path diagram from Example 2). The key difference is that the path coefficients are just a tool for linking incomplete variables and do not represent a substantive theory.

Blimp Script

The code block below shows Blimp script `Ex11.1.inp`. The first five lines can be viewed as a set of commands that specify information about the data and variables. The `DATA` command specifies the name of the input text file. No file path is required when the data file is located in the same directory as the script, as it is here. Starting on line 2, the `VARIABLES` command names the data columns. The `ORDINAL` command on line 4 identifies binary and ordinal variables. Binary variables can be defined as ordinal or nominal, as the statistical models are identical. The `MISSING` command on line 5 defines a global missing value code as 999.

Blimp Script Ex11.1.imp

```
1   DATA: mathachievement.dat;
2   VARIABLES: id condition male frlunch atrisk stanread efficacy anxiety
3      mathpre mathpost;
4   ORDINAL: frlunch atrisk efficacy;
5   MISSING: 999;
6   FIXED: mathpre;
7   CENTER: anxiety;
8   MODEL:
9   focal.model:
10  mathpost ~ anxiety anxiety^2@beta2 frlunch mathpre efficacy;
11  auxiliary.models:
12  stanread atrisk ~ mathpost anxiety frlunch efficacy mathpre;
13  TEST:
14  beta2 = 0;
15  SEED: 90291;
16  BURN: 10000;
17  ITERATIONS: 10000;
```

The `FIXED`, `CENTER`, `MODEL`, and `TEST` blocks can be viewed as a set. The `FIXED` command identifies a complete predictor, which does not require a distribution or regression model. The `CENTER` command deviates anxiety scores (the variable with the non-linear term) at their iteratively-estimated grand mean. Beginning on line 8, the `MODEL` command lists the regression models, with outcome variables to the left of the tilde and predictors to the right. The code uses labels (`focal.model` and `auxiliary.models`) to order output tables, such that the focal model appears first followed by the auxiliary variable models. The focal model listed on line 10 includes a squared term, which is specified by appending `^2` to the variable name. The quadratic slope coefficient is labeled using the @ symbol. Blimp automatically configures the explanatory variable models under the assumption that they are normally distributed. Line 12 is a syntax shortcut that produces the two auxiliary variable regression models in Equation 25; in the first model, `READ2` is regressed on the focal variables, and the second model features `STANREAD7` regressed on `READ2` and the focal variables. The `TEST` command uses the parameter labels to specify a custom hypothesis test that the quadratic effect equals zero. This command produces the Bayesian Wald test (Asparouhov & Muthén, 2021), which is essentially a chi-square statistic that captures the discrepancy between the Bayesian point estimates (posterior means) and the hypothesized value of zero.

Finally, lines 15 through 17 can be viewed as a block of commands that specify features of the MCMC algorithm: the `SEED` command gives an integer string that initializes the random number generator, the `BURN` command specifies the number of iterations for the warm-up or burn-in period, and the `ITERATIONS` command gives the number of MCMC iterations on which the analysis summaries are based (essentially, the number of MCMC cycles following the warm-up period).

Blimp Output

Prior to inspecting the parameter estimates, it is important to investigate the potential scale reduction (PSR) factor diagnostics (Gelman & Rubin, 1992) to determine whether MCMC has converged. Blimp divides the burn-in period into 20 equal segments, and it computes the PSR diagnostic for every parameter. The table located near the top of the output reports the highest (worst) PSR value across all parameters in every model. A common recommendation is that these values should be less than 1.05 or perhaps 1.10 (Asparouhov & Muthén, 2010a; Gelman et al., 2014). If the PSR in the bottom row of the table (the final check of the burn-in period) is above these cutoffs, then rerun the analysis with a longer burn-in period.

```
BURN-IN POTENTIAL SCALE REDUCTION (PSR) OUTPUT:

  NOTE: Split chain PSR is being used. This splits each chain's
        iterations to create twice as many chains.

  Comparing iterations across 2 chains     Highest PSR   Parameter #
                           251 to 500            1.045            61
                           501 to 1000           1.087            58
                           751 to 1500           1.042            58
                          1001 to 2000           1.065            59
                                   ...             ...            ..
                          4501 to 9000           1.045            60
                          4751 to 9500           1.009            58
                          5001 to 10000          1.012            59
```

The next section of the output displays information about the variables in the analysis and the models used for estimation. This output table mimics the one from Example 6. In the interest of space, we point readers to that example for additional details.

Thus far, the examples have generally used 95% credible intervals to evaluate the significance of individual slope coefficients. The `TEST` command in the previous script requested a Bayesian Wald chi-square statistic (Asparouhov & Muthén, 2021) that evaluates the null hypothesis that the population quadratic effect equals zero. The chi-square statistic, degrees of freedom, and *p*-value appear near the bottom of the `MODEL FIT` section under the `WALD TEST` heading. The test statistic is statistically significant, thus refuting the null hypothesis.

```
MODEL FIT:

  ...

  WALD TESTS (Asparouhov & Muthén, 2021)

  Test #1

    Full:
      [1]  mathpost ~ Intercept anxiety frlunch efficacy mathpre anxiety^2@beta2

    Restricted:
      [1]  mathpost ~ Intercept anxiety frlunch efficacy mathpre anxiety^2@beta2

    Constraints in Restricted:
      [1]  beta2 = 0

    Wald Statistic (Chi-Square)                11.705
    Number of Parameters Tested (df)                1
    Probability                                 0.001
```

The table summarizing the focal regression model is shown below. The table includes unstandardized coefficients, standardized slopes, and variance explained effect size estimates. MCMC estimation produces a distribution for each parameter in the table. The median and standard deviation columns describe the center and spread of the posterior distributions; although they make no reference to drawing repeated samples, they are analogous—and numerically equivalent in most cases—to frequentist point estimates and standard errors. The 95% credible intervals in the rightmost columns give a range that captures 95% of the parameter's distribution. These are akin to confidence intervals, but the intervals describe parameter distributions rather than characteristics of repeated samples. The `N_Eff` values in

rightmost column of the table give the effective number of MCMC samples for each parameter. These quantities essentially represent the number of independent estimates on which the parameter summaries are based after removing autocorrelations from the MCMC process. Gelman et al. (2014, p. 287) recommend values greater than 100. All values in the example table exceed this recommended minimum. In cases where the `N_Eff` values are insufficient, increasing the value on the `ITERATIONS` command will remedy the issue.

```
OUTCOME MODEL ESTIMATES:

  Summaries based on 10000 iterations using 2 chains.

  focal.model block:

Outcome Variable:  mathpost

Grand Mean Centered: anxiety


Parameters                      Median     StdDev       2.5%      97.5%        PSR      N_Eff
                               ------------------------------------------------------------------
Variances:
  Residual Var.                 50.966      5.388     41.845     63.021      1.000   5413.246

Coefficients:
  Intercept                     32.689      3.457     25.968     39.571      1.000   6410.684
  anxiety                        0.041      0.084     -0.121      0.206      1.000   4221.113
  frlunch                       -5.840      1.073     -7.948     -3.750      1.000   4993.497
  efficacy                       1.103      0.342      0.444      1.781      1.000   5838.380
  mathpre                        0.471      0.067      0.338      0.602      1.000   6319.544
  anxiety^2                     -0.021      0.006     -0.033     -0.009      1.000   5091.389

Standardized Coefficients:
  anxiety                        0.031      0.064     -0.093      0.157      1.000   4228.247
  frlunch                       -0.298      0.051     -0.394     -0.195      1.000   5053.969
  efficacy                       0.184      0.056      0.074      0.293      1.000   5698.632
  mathpre                        0.423      0.055      0.310      0.523      1.000   6522.033
  anxiety^2                     -0.204      0.058     -0.315     -0.087      1.000   5054.272

Proportion Variance Explained
  by Coefficients                0.453      0.045      0.359      0.539      1.000   5780.687
  by Residual Variation          0.547      0.045      0.462      0.641      1.000   5780.687

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

In a curvilinear regression model, the lower-order term for math anxiety is a conditional effect that depends on scaling or centering. The slope conveys the instantaneous linear change in the outcome at the anxiety mean, controlling for all other predictors (β_1 = 0.04). The negative

quadratic coefficient ($\beta_2 = -0.02$) indicates that the positive association at the mean decreases (i.e., becomes less positive) as anxiety increases (and vice versa). At high enough levels of anxiety, the association becomes negative, such that anxiety has a debilitating effect on math performance.

The Blimp output also includes tables of regression model parameters for the auxiliary variables as well as the auto-generated models for incomplete predictors. These additionally results are not of substantive interest and would not be reported. The auxiliary variable models appear in `OUTCOME MODEL ESTIMATES` section with the focal results, and the auto-generated predictor models are displayed under the heading `PREDICTOR MODEL ESTIMATES`.

Saving Multiple Imputations

MCMC estimation imputes missing values at every iteration, such that the resulting Bayesian estimates average over thousands of plausible replacement scores (10,000 sets in this example). A subset of the imputations can be saved for reanalysis in the frequentist framework, if desired. The Blimp input file `Ex11.2.imp` is identical `Ex11.1.imp`, but it adds the following lines at the bottom of the script.

```
NIMPS: 20;
CHAINS: 20;
SAVE:
stacked = ./imps/imps.dat;
separate = ./imps/imp*.dat;
```

The `NIMPS`, `CHAINS` and `SAVE` commands can be viewed as a set. Setting `NIMPS` equal to `CHAINS` saves a single filled-in data set from the final iteration of a unique `MCMC` process, thus avoiding autocorrelation among the imputations. The `SAVE` command provides a name for the imputed data sets. The script illustrates how to save data sets in two common formats. The stacked keyword creates a stacked file where all imputations are in a single file, and the separate keyword saves each imputed data set to a separate file with the asterisk replaced by a numeric index. To keep things organized, the `./imps` part of the file path points to a subfolder named `imps` located within the same folder as the script and data. The separate keyword also creates a list of file names needed for analysis in Mplus (in this example, a filed called `implist.dat` located in the `imps` folder).

When saving imputations, the bottom of the Blimp output file displays a table listing the order of the variables in the output data sets. All variables are saved regardless of whether they appeared in the fitted models. When saving data to a stacked file (e.g., for analysis in R or other packages), the first variable in the file is an integer index that identifies which data set each row belongs to (e.g., an integer variable that ranges from 1 to 20 in this example).

```
VARIABLE ORDER IN IMPUTED DATA:

  separate = './imps/imp*.dat'

     id condition male frlunch atrisk stanread efficacy anxiety
     mathpre mathpost

   stacked = './imps/imps.dat'

     imp# id condition male frlunch atrisk stanread efficacy
     anxiety mathpre mathpost
```

The imputed data sets are subsequently analyzed in another software package, and estimates and standard errors are combined using Rubin's rules (Little & Rubin, 2020). The analysis phase does not utilize the auxiliary variables, as their information is embedded in the imputations. Scripts for analyzing the imputed data sets are found in the next subsections.

Analyzing Imputations in Mplus

In lieu of the Bayesian estimates, Blimp's `SAVE` command can be used to save multiple imputations for analysis in the frequentist framework. Returning to the previous Blimp script, the `SAVE` command and the `separate` keyword saved each imputed data set to a separate file with the asterisk replaced by a numeric index. The separate keyword also creates a list of file names needed for analysis in Mplus (in this example, a filed called `implist.dat` located in the `imps` subfolder). Example 6 shows the contents of this file.

The Mplus input file for analyzing the imputations is `Ex11.inp`. The major commands are similar to the `Ex8.inp` file described in Example 8. Consistent with previous multiple imputation analysis scripts, the `DATA` command lists the text file containing the names of the imputed data sets (the `implist.dat` file located in the `./imps` subdirectory). The `type =`

`imputation` subcommand instructs Mplus that the input data is a list of file names. Second, the missing subcommand is omitted because the analysis variables are now complete. Finally, the `MODEL` section no longer specifies a normal distribution for the predictors or models for the auxiliary variables.

The script also invokes one new feature. On line 9, the `center` subcommand under the `DEFINE` command centers anxiety scores at their grand mean. Line 10 then computes a new variable equal to the square of the centered predictor. Importantly, new variables computed with the `DEFINE` command must appear at the end of the `usevariables` list on line 7. The script is shown below.

Mplus Script Ex11.inp

```
1   DATA:
2   file = ./imps/implist.dat;
3   type = imputation;
4   VARIABLE:
5   names = id condition male frlunch atrisk stanread
6     efficacy anxiety mathpre mathpost;
7   usevariables = mathpost anxiety frlunch efficacy mathpre anxietysq;
8   DEFINE:
9   center anxiety (grandmean);
10  anxietysq = anxiety^2;
11  MODEL:
12  mathpost on anxiety anxietysq frlunch efficacy mathpre;
13  OUTPUT:
14  stdyx cinterval;
```

Mplus Output

The table of unstandardized parameter estimates is shown below. The first two columns display the pooled unstandardized estimates and standard errors, and the third and fourth columns display the corresponding z-statistics and p-values. The focal model results are shown in bold typeface. The `Rate of Missing` column (also called the fraction of missing information in the literature) quantifies the imputation noise in each estimate as proportion of its squared standard error.

```
MODEL RESULTS

                                                    Two-Tailed   Rate of
                  Estimate       S.E.  Est./S.E.    P-Value    Missing

 MATHPOST ON
    ANXIETY          0.034      0.083      0.407      0.684      0.304
    ANXIETYSQ       -0.021      0.006     -3.652      0.000      0.187
    FRLUNCH         -5.687      1.138     -4.998      0.000      0.371
    EFFICACY         1.052      0.341      3.085      0.002      0.276
    MATHPRE          0.471      0.063      7.414      0.000      0.142

 Intercepts
    MATHPOST        32.944      3.283     10.035      0.000      0.135

 Residual Variances
    MATHPOST        49.149      5.588      8.795      0.000      0.388
```

In a curvilinear regression model, the lower-order term for math anxiety is a conditional effect that depends on scaling or centering. The slope conveys the instantaneous linear change in the outcome at the anxiety mean, controlling for all other predictors ($\hat{\beta}_1 = 0.03$). The negative quadratic coefficient ($\hat{\beta}_2 = -0.02$) indicates that the positive association at the mean decreases (i.e., becomes less positive) as anxiety increases (and vice versa). At high enough levels of anxiety, the association becomes negative, such that anxiety has a debilitating effect on math performance. The output also includes a table with standardized coefficients and the *R*-squared statistic. Note that these estimates are numerically identical to those from Bayesian and maximum likelihood estimation.

Analyzing Imputations in R

Returning to the previous Blimp script, the `SAVE` command and the `stacked` keyword saved the imputed data sets to a single stacked file with an index variable in the first column identifying the individual files. The stacked file is appropriate for analyzing data in R, SAS, SPSS, and Stata, among others.

The R input file for the analysis is `Ex11.R`. The code block below shows the commands that import the data.

R Script Ex11.R

```
1   library(fdir)
2   library(mitml)
3   set()
4   imps <- read.table("./imps/imps.dat")
5   names(imps) <- c("imputation", "id", "condition","male","frlunch",
6     "atrisk", "stanread","efficacy", "anxiety", "mathpre", "mathpost")
7   imps$anxiety.cgm <- imps$anxiety - mean(imps$anxiety)
8   imps$anxiety.sq <- imps$anxiety.cgm^2
```

The example requires the `fdir` and `mitml` packages, which are loaded on lines 1 and 2. On line 3, the `set()` function of the `fdir` package identifies the file path of the folder containing the R script and sets this location as the working directory. On line 4, the `read.table` command imports the stacked data. It is only necessary to specify the name of the input data file. No file path is required when the data reside in the same folder as the R script as is the case here. Importantly, the first variable named `IMPUTATION` is the index that identifies the individual files. Finally, line 7 creates a new centered variable called `ANXIETY.CGM`, and line 8 computes the square of the centered variable.

The next block of code relies on the `mitml` package to fit the model to each data set and pool the results using Rubin's rules.

R Script Ex11.R, continued

```
9   implist <- as.mitml.list(split(imps, imps$imputation))
10  fit <- with(implist, lm(mathpost ~ anxiety.cgm + anxiety.sq +
11    frlunch  + efficacy + mathpre))
12  estimates <- testEstimates(fit, extra.pars = T, df.com = 244)
13  estimates
14  confint(estimates)
```

The `implist` command on line 9 unstacks the data and creates a list that contains the individual files. Line 10 fits the focal regression model using the `lm` function, and line 12 uses the `testEstimates` function in `mitml` to implement Rubin's pooling rules and save the results in

an object called `estimates`. The `df.com` parameter is the denominator degrees of freedom that would have resulted had there been no missing data (i.e., $N–K–1$ degrees of freedom, where K is the number of predictors). This argument produces Barnard and Rubin degrees of freedom values. Finally, lines 13 and 14 print the estimates and confidence intervals.

R Output

The table of unstandardized pooled parameter estimates is shown below. The first two columns display the pooled unstandardized estimates and standard errors, and the third through fifth columns display the corresponding test statistics. The focal model results are shown in bold typeface. The `RIV` column (relative increase in variance) is a fraction comparing imputation noise to complete-data sampling variation, and the `FMI` column (fraction of missing information in the literature) quantifies the imputation noise in each estimate as proportion of its squared standard error.

```
> estimates

Call:

testEstimates(model = fit, extra.pars = T, df.com = 244)

Final parameter estimates and inferences obtained from 20 imputed data sets.

             Estimate Std.Error   t.value        df   P(>|t|)       RIV       FMI
(Intercept)    32.943     3.318     9.929   177.001     0.000     0.150     0.140
anxiety.cgm     0.034     0.084     0.404    96.483     0.687     0.415     0.307
anxiety.sq     -0.021     0.006    -3.616   147.802     0.000     0.221     0.192
frlunch        -5.687     1.147    -4.960    76.330     0.000     0.554     0.373
efficacy        1.052     0.344     3.058   107.170     0.003     0.361     0.279
mathpre         0.471     0.064     7.336   173.143     0.000     0.159     0.147

                   Estimate
Residual~~Residual   50.358

Hypothesis test adjusted for small samples with df=[244]
complete-data degrees of freedom.
```

```
> confint(estimates)
                  2.5 %       97.5 %
(Intercept) 26.39566909 39.490858174
anxiety.cgm -0.13268668  0.200544222
anxiety.sq  -0.03287417 -0.009639516
frlunch     -7.97089460 -3.403528840
efficacy     0.37008949  1.734175237
mathpre      0.34392328  0.597097949
```

In a curvilinear regression model, the lower-order term for math anxiety is a conditional effect that depends on scaling or centering. The slope conveys the instantaneous linear change in the outcome at the anxiety mean, controlling for all other predictors ($\hat{\beta}_1 = 0.03$). The negative quadratic coefficient ($\hat{\beta}_2 = -0.02$) indicates that the positive association at the mean decreases (i.e., becomes less positive) as anxiety increases (and vice versa). At high enough levels of anxiety, the association becomes negative, such that anxiety has a debilitating effect on math performance. Note that these estimates are numerically identical to those from Bayesian and maximum likelihood estimation.

SECTION 3: MODEL-AGNOSTIC MULTIPLE IMPUTATION

EXAMPLE 12: FULLY CONDITIONAL SPECIFICATION IMPUTATION FOR A PAIRED-SAMPLES COMPARISON

This example illustrates model-agnostic fully conditional specification multiple imputation for a paired-samples test involving pretest and posttest scores. The analysis uses the `mathachievement.dat` data set taken from an educational intervention where 250 students were assigned to an intervention and comparison condition. The file includes pretest and posttest math achievement scores, a measure of math self-efficacy, standardized reading scores taken from a statewide assessment, and several sociodemographic variables. The analysis variables are as follows.

Name	Definition	Missing %	Scale
	Focal Variables		
MATHPRE	Math achievement pretest	0	Numeric
MATHPOST	Math achievement posttest	18.0	Numeric
	Auxiliary Variables		
FRLUNCH	Lunch assistance code	4.4	0 = None, 1 = Free/reduced lunch
STANREAD	Standardized reading	9.2	Numeric
EFFICACY	Math self-efficacy rating	9.6	Ordinal (1 to 6)

Imputation and Analysis Models

Fully conditional specification uses a sequence of regression models to fill in missing values. Specifically, each MCMC iteration fits a series of models where one incomplete variable is regressed on all other variables. The predicted values and residual variance from each model define the center and spread of the imputed values, which are drawn at random from a normal distribution. After imputing the missing scores, the filled-in variable becomes a predictor in all other imputation models in the sequence. The imputation stage should include all variables and effects for the subsequent analyses, and it should incorporate auxiliary variables that either predict missingness or correlate with the incomplete variables (Collins et al., 2001).

A common goal of model-agnostic imputation is to generate imputations for different purposes (e.g., descriptive summaries, several analyses within the same project). To illustrate an entire multiple imputation analysis, suppose that one use of the filled-in data sets involves a

paired-samples test of the changes between pretest and posttest. The analysis can be cast as an empty regression model with change scores as the outcome variable.

$$CHANGE = \beta_0 + \varepsilon \tag{26}$$

The variable `CHANGE` is computed as `MATHPOST` minus `MATHPRE`.

Blimp Script

The code block below shows Blimp script `Ex12.inp`.

Blimp Script Ex12.imp

```
1   DATA: mathachievement.dat;
2   VARIABLES: id condition male frlunch atrisk stanread efficacy anxiety
3      mathpre mathpost;
4   ORDINAL: frlunch efficacy;
5   MISSING: 999;
6   FIXED: mathpre;
7   FCS: mathpost mathpre frlunch stanread efficacy;
8   SEED: 90291;
9   BURN: 5000;
10  ITERATIONS: 10000;
11  NIMPS: 20;
12  CHAINS: 20;
13  SAVE:
14  stacked = ./imps/imps.dat;
15  separate = ./imps/imp*.dat;
```

The first five lines can be viewed as a set of commands that specify information about the data and variables. The `DATA` command specifies the name of the input text file. No file path is required when the data file is located in the same directory as the script, as it is here. Starting on line 2, the `VARIABLES` command names the data columns. The `ORDINAL` command on line 4 identifies a pair of binary variables. Binary variables can alternatively be identified using the `NOMINAL` command because the underlying statistical models are identical. Finally, the `MISSING` command on line 5 defines a global missing value code as 999.

Next, the FCS command lists all variables—complete or incomplete—that are included in the imputation phase. The FIXED command identifies a complete variable that does not require imputation. This reduces computational time because complete variables do not require a regression model. Lines 8 through 10 can also be viewed as a block of commands that specify features of the MCMC algorithm: the SEED command gives an integer string that initializes the random number generator, the BURN command specifies the number of iterations for the warm-up or burn-in period, and the ITERATIONS command gives the number of MCMC iterations on which the imputation model summaries are based (essentially, the total number of MCMC cycles across all chains following the warm-up period).

The NIMPS, CHAINS and SAVE commands can be viewed as a set. Setting NIMPS equal to CHAINS saves a single filled-in data set from the final iteration of a unique MCMC process, thus avoiding autocorrelation among the imputations. The SAVE command provides a name for the imputed data sets. The script illustrates how to save data sets in two common formats. The stacked keyword creates a stacked file where all imputations are in a single file, and the separate keyword saves each imputed data set to a separate file with the asterisk replaced by a numeric index. To keep things organized, the ./imps part of the file path points to a subfolder named imps located within the same folder as the script and data. The separate keyword also creates a list of file names needed for analysis in Mplus (in this example, a filed called implist.dat located in the imps folder).

Blimp Output

Prior to inspecting the parameter estimates, it is important to investigate the potential scale reduction (PSR) factor diagnostics (Gelman & Rubin, 1992) to determine whether MCMC has converged. Blimp divides the burn-in period into 20 equal segments, and it computes the PSR diagnostic for every parameter. The table located near the top of the output reports the highest (worst) PSR value across all parameters in every model. A common recommendation is that these values should be less than 1.05 or perhaps 1.10 (Asparouhov & Muthén, 2010a; Gelman et al., 2014). If the PSR in the bottom row of the table (the final check of the burn-in period) is above these cutoffs, then rerun the analysis with a longer burn-in period.

```
BURN-IN POTENTIAL SCALE REDUCTION (PSR) OUTPUT:

  NOTE: Split chain PSR is being used. This splits each chain's
        iterations to create twice as many chains.

  Comparing iterations across 2 chains     Highest PSR   Parameter #
                          126 to 250             1.480            23
                          251 to 500             1.395            24
                          376 to 750             1.272            23
                          501 to 1000            1.250            23
                          626 to 1250            1.163            23
                          751 to 1500            1.144            23
                          876 to 1750            1.093            23
                         1001 to 2000            1.101            23
                         1126 to 2250            1.140            23
                         1251 to 2500            1.097            24
                         1376 to 2750            1.104            23
                         1501 to 3000            1.116            23
                         1626 to 3250            1.101            23
                         1751 to 3500            1.075            23
                         1876 to 3750            1.068            23
                         2001 to 4000            1.076            23
                         2126 to 4250            1.055            23
                         2251 to 4500            1.048            22
                         2376 to 4750            1.038            23
                         2501 to 5000            1.042            23
```

The next output excerpt shows information about the data and the variables in the imputation models.

```
DATA INFORMATION:

  Sample Size:              250
  Missing Data Rates:

                  frlunch = 04.40
                 stanread = 09.20
                 efficacy = 09.60
                 mathpost = 18.00

VARIABLES IN IMPUTATION MODEL:

  Fixed variables:        mathpre
  Incomplete continuous:  stanread mathpost
  Incomplete ordinal:     frlunch efficacy

  NUMBER OF PARAMETERS
  Imputation Models:      26
```

MCMC estimation produces a distribution for each parameter in every unique imputation model. The median and standard deviation columns describe the center and spread of the posterior distributions; although they make no reference to drawing repeated samples, they are analogous—and numerically equivalent in most cases—to frequentist point estimates and standard errors. The 95% credible intervals in the rightmost columns give a range that captures 95% of the parameter's distribution. These are akin to confidence intervals, but the intervals describe parameter distributions rather than characteristics of repeated samples. The `N_Eff` values in rightmost column of the table give the effective number of MCMC samples for each parameter. These quantities essentially represent the number of independent estimates on which the parameter summaries are based after removing autocorrelations from the MCMC process. Gelman et al. (2014, p. 287) recommend values greater than 100. All values in the example table exceed this recommended minimum. In cases where the `N_Eff` values are insufficient, increasing the value on the `ITERATIONS` command will remedy the issue.

The Blimp output includes tables of regression parameters for every incomplete variable's imputation model. The imputation model parameters are not of substantive interest and would not be reported. An example table is shown below.

```
Missing variable:   mathpost

Parameters                          Median     StdDev       2.5%      97.5%        PSR      N_Eff
                                ------------------------------------------------------------------

Grand Mean                          56.504      0.555     55.397     57.567      1.005   4353.468

Level 1:
  frlunch                           -2.188      0.575     -3.270     -1.038      1.006   3351.406
  stanread                           0.189      0.060      0.073      0.306      1.004   4430.891
  efficacy                           1.222      0.517      0.216      2.252      1.005   4406.012
  mathpre                            0.476      0.062      0.354      0.596      1.002   6043.712
  Residual Var.                     45.242      4.994     36.454     56.257      1.004   4585.610

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

When saving imputations, the bottom of the Blimp output file displays a table listing the order of the variables in the output data sets. All variables are saved regardless of whether they appeared in the fitted models. When saving data to a stacked file (e.g., for analysis in R or other packages), the first variable in the file is an integer index that identifies which data set each row belongs to (e.g., an integer variable that ranges from 1 to 20 in this example).

```
VARIABLE ORDER IN IMPUTED DATA:

  separate = './imps/imp*.dat'

     id condition male frlunch atrisk stanread efficacy anxiety
     mathpre mathpost

   stacked = './imps/imps.dat'

     imp# id condition male frlunch atrisk stanread efficacy
     anxiety mathpre mathpost
```

The imputed data sets are subsequently analyzed in another software package, and estimates and standard errors are combined using Rubin's rules (Little & Rubin, 2020). The analysis phase does not utilize the auxiliary variables, as their information is embedded in the imputations. Scripts for analyzing the imputed data sets are found in the next subsections.

Analyzing Imputations in Mplus

Returning to the previous Blimp script, the `SAVE` command and the `separate` keyword saved each imputed data set to a separate file with the asterisk replaced by a numeric index. The separate keyword also creates a list of file names needed for analysis in Mplus (in this example, a filed called `implist.dat` located in the `imps` subfolder). The contents of this file are as follows.

```
imp1.dat
imp2.dat
imp3.dat
imp4.dat
imp5.dat
imp6.dat
imp7.dat
imp8.dat
imp9.dat
imp10.dat
imp11.dat
imp12.dat
imp13.dat
imp14.dat
imp15.dat
imp16.dat
imp17.dat
imp18.dat
imp19.dat
imp20.dat
```

The Mplus input file for analyzing the imputations is `Ex12.inp`. Following previous imputation analysis examples, the `DATA` command lists the text file containing the names of the imputed data sets (the `implist.dat` file located in the `./imps` subdirectory). The `type = imputation` subcommand instructs Mplus that the input data is a list of file names. The `usevariables` subcommand of the `VARIABLE` command selects variables for the analysis. The `DEFINE` command beginning on line 8 computes the change or difference score by subtracting the pretest from posttest. Importantly, new variables computed with the `DEFINE` command must appear at the end of the `usevariables` list on line 7. In this example, the new change score is the only variable in the model. Listing the change score variable in the `MODEL` section estimates

the mean and variance of the variable. Finally, listing the `cinterval` keyword after `OPTION` prints confidence intervals. The code block below shows the analysis and pooling script.

Mplus Script Ex12.inp

```
DATA:
file = ./imps/implist.dat;
type = imputation;
VARIABLE:
names = id condition male frlunch lowach stanread efficacy
  anxiety mathpre mathpost;
usevariables =  change;
DEFINE:
change = mathpost - mathpre;
MODEL:
change;
OUTPUT:
cinterval;
```

Mplus Output

The table of unstandardized parameter estimates is shown below. The first two columns display the pooled unstandardized estimates and standard errors, and the third and fourth columns display the corresponding z-statistics and p-values. The focal model results are shown in bold typeface. The `Rate of Missing` column (also called the fraction of missing information in the literature) quantifies the imputation noise in each estimate as proportion of its squared standard error.

```
MODEL RESULTS

                                                    Two-Tailed   Rate of
                  Estimate       S.E.  Est./S.E.    P-Value    Missing

 Means
    CHANGE           6.418      0.577     11.131      0.000      0.130

 Variances
    CHANGE          72.439      6.864     10.554      0.000      0.109
```

The results are interpreted in the same way as a complete-data paired-samples test. For example, the intercept represents the mean change from pretest to posttest. The corresponding test statistic indicates that the change is statistically different from zero ($z = 11.13$, $p < .001$).

Analyzing Imputations in R

Returning to the previous Blimp script, the `SAVE` command and the `stacked` keyword saved the imputed data sets to a single stacked file with an index variable in the first column identifying the individual files. The stacked file is appropriate for analyzing data in R, SAS, SPSS, and Stata, among others. The R input file for the analysis is `Ex12.R`. The code block below shows the commands that import the data.

R Script Ex12.R

```
1   library(fdir)
2   library(mitml)
3   set()
4   imps <- read.table("./imps/imps.dat")
5   names(imps) <- c("imputation", "id", "condition","male","frlunch",
6     "atrisk", "stanread","efficacy", "anxiety", "mathpre", "mathpost")
7   imps$change <- imps$mathpost - imps$mathpre
```

The example requires the `fdir` and `mitml` packages, which are loaded on lines 1 and 2. On line 3, the `set()` function of the `fdir` package identifies the file path of the folder containing the

R script and sets this location as the working directory. On line 4, the `read.table` command imports the stacked data. It is only necessary to specify the name of the input data file. No file path is required when the data reside in the same folder as the R script as is the case here. Next, variable names are listed beginning on line 5. Importantly, the first variable named `IMPUTATION` is the index that identifies the individual files. Finally, line 7 computes a new variable named `CHANGE` that represents the gain from pretest to posttest.

The next block of code relies on the `mitml` package to fit the model to each data set and pool the results using Rubin's rules. The `implist` command on line 9 unstacks the data and creates a list that contains the individual files. Line 10 fits the focal regression model using the `lm` function, and line 11 uses the `testEstimates` function in `mitml` to implement Rubin's pooling rules and save the results in an object called `estimates`. The `df.com` parameter is the denominator degrees of freedom that would have resulted had there been no missing data (i.e., $N–K–1$ degrees of freedom, where K is the number of predictors). This argument produces Barnard and Rubin degrees of freedom values. Finally, lines 12 and 13 print the estimates and confidence intervals.

R Script Ex12.R, continued

```
 9  implist <- as.mitml.list(split(imps, imps$imputation))
10  fit <- with(implist, lm(change ~ 1))
11  estimates <- testEstimates(fit, extra.pars = T, df.com = 249)
12  estimates
13  confint(estimates)
```

R Output

The table of unstandardized pooled parameter estimates is shown below. The first two columns display the pooled unstandardized estimates and standard errors, and the third through fifth columns display the corresponding test statistics. The focal model results are shown in bold typeface. The `RIV` column (relative increase in variance) is a fraction comparing imputation noise to complete-data sampling variation, and the `FMI` column (fraction of missing information in the literature) quantifies the imputation noise in each estimate as proportion of its squared standard error.

```
> estimates

Call:

testEstimates(model = fit, extra.pars = T, df.com = 249)

Final parameter estimates and inferences obtained from 20 imputed data sets.

             Estimate Std.Error   t.value        df   P(>|t|)       RIV       FMI
(Intercept)     6.418     0.578    11.112   181.675     0.000     0.147     0.137

                   Estimate
Residual~~Residual   72.732

Hypothesis test adjusted for small samples with df=[249]
complete-data degrees of freedom.

> confint(estimates)
               2.5 %  97.5 %
(Intercept) 5.278415 7.55776
```

The results are interpreted in the same way as a complete-data paired-samples test. For example, the intercept represents the mean change from pretest to posttest. The corresponding test statistic indicates that the change is statistically different from zero ($t = 11.11$, $p < .001$).

EXAMPLE 13: FULLY CONDITIONAL SPECIFICATION IMPUTATION WITH MULTIVARIATE NORMAL DATA

This example illustrates model-agnostic fully conditional specification multiple imputation with multivariate normal data. The analysis uses the `behaviorachievement.dat` data set taken from a longitudinal study that followed 138 students from primary through middle school. The file includes three annual assessments of broad reading and math achievement beginning in the first grade, seventh grade standardized achievement test scores taken from a statewide assessment, and a final measure of broad reading and math obtained in ninth grade. The data also contain teacher ratings of behavioral symptoms and learning problems were also obtained in the first grade. The data description at the beginning of this document provides additional details. The variables for this analysis are as follows.

Name	Definition	Missing %	Scale
	Focal Variables		
BEHSYMP1	1st grade behavioral symptoms	3.6	Numeric
LRNPROB1	1st grade learning problems	2.2	Numeric
READ1	1st grade broad reading composite	6.5	Numeric
READ9	9th grade broad reading composite	17.4	Numeric
	Auxiliary Variables		
READ2	2nd grade broad reading composite	9.4	Numeric
STANREAD7	7th grade standardized math	19.6	Numeric

Imputation and Analysis Models

Fully conditional specification uses a sequence of regression models to fill in missing values. Specifically, each MCMC iteration fits a series of models where one incomplete variable is regressed on all other variables. The predicted values and residual variance from each model define the center and spread of the imputed values, which are drawn at random from a normal distribution. After imputing the missing scores, the filled-in variable becomes a predictor in all other imputation models in the sequence. The imputation stage should include all variables and effects for the subsequent analyses, and it should incorporate auxiliary variables that either predict missingness or correlate with the incomplete variables (Collins et al., 2001).

A common goal of model-agnostic imputation is to generate imputations for different purposes (e.g., descriptive summaries, several analyses within the same project). To illustrate an entire multiple imputation analysis, suppose that one use of the filled-in data sets involves a model where ninth grade broad reading scores are regressed on first grade reading achievement and teacher-rated learning problems and behavioral symptoms.

$$READ_9 = \beta_0 + \beta_1(READ_1) + \beta_2(LRNPROB_1) + \beta_3(BEHSYMP_1) + \varepsilon \tag{27}$$

Examples 1 and 6 used the same analysis model to illustrate maximum likelihood estimation, Bayesian estimation, and model-based multiple imputation.

Blimp Script

The code block below shows Blimp script `Ex13.inp`.

Blimp Script Ex13.imp

```
DATA: behaviorachievement.dat;
VARIABLES: id male hispanic riskgrp atrisk behsymp1 lrnprob1
  read1 read2 read3 read9 read9grp stanread7
  math1 math2 math3 math9 math9grp stanmath7;
MISSING: 999;
FCS: read9 read1 lrnprob1 behsymp1 stanread7 read2;
SEED: 90291;
BURN: 2000;
ITERATIONS: 10000;
NIMPS: 20;
CHAINS: 20;
SAVE:
stacked = ./imps/imps.dat;
separate = ./imps/imp*.dat;
```

The first five lines can be viewed as a set of commands that specify information about the data and variables. The `DATA` command specifies the name of the input text file. No file path is required when the data file is located in the same directory as the script, as it is here. Starting on

line 2, the `VARIABLES` command names the data columns, and the `MISSING` command on line 5 defines a global missing value code as 999.

Next, the `FCS` command lists all variables—complete or incomplete—that are included in the imputation phase. Using the `FIXED` command to identify complete variables reduces computational time because these variables do not require a regression model (see Example 12). Lines 7 through 9 can also be viewed as a block of commands that specify features of the MCMC algorithm: the `SEED` command gives an integer string that initializes the random number generator, the `BURN` command specifies the number of iterations for the warm-up or burn-in period, and the `ITERATIONS` command gives the number of MCMC iterations on which the imputation model summaries are based (essentially, the total number of MCMC cycles across all chains following the warm-up period).

The `NIMPS`, `CHAINS` and `SAVE` commands can be viewed as a set. Setting `NIMPS` equal to `CHAINS` saves a single filled-in data set from the final iteration of a unique `MCMC` process, thus avoiding autocorrelation among the imputations. The `SAVE` command provides a name for the imputed data sets. The script illustrates how to save data sets in two common formats. The stacked keyword creates a stacked file where all imputations are in a single file, and the separate keyword saves each imputed data set to a separate file with the asterisk replaced by a numeric index. To keep things organized, the `./imps` part of the file path points to a subfolder named `imps` located within the same folder as the script and data. The separate keyword also creates a list of file names needed for analysis in Mplus (in this example, a filed called `implist.dat` located in the `imps` folder).

Blimp Output

Prior to inspecting the parameter estimates, it is important to investigate the potential scale reduction (PSR) factor diagnostics (Gelman & Rubin, 1992) to determine whether MCMC has converged. Blimp divides the burn-in period into 20 equal segments, and it computes the PSR diagnostic for every parameter. The table located near the top of the output reports the highest (worst) PSR value across all parameters in every model. A common recommendation is that these values should be less than 1.05 or perhaps 1.10 (Asparouhov & Muthén, 2010a; Gelman et al., 2014). If the PSR in the bottom row of the table (the final check of the burn-in period) is above these cutoffs, then rerun the analysis with a longer burn-in period.

```
BURN-IN POTENTIAL SCALE REDUCTION (PSR) OUTPUT:

  NOTE: Split chain PSR is being used. This splits each chain's
        iterations to create twice as many chains.

  Comparing iterations across 2 chains     Highest PSR   Parameter #
                            51 to 100            1.436            22
                           101 to 200            1.245            22
                           151 to 300            1.132            22
                           201 to 400            1.094            22
                           251 to 500            1.070            22
                           301 to 600            1.063            22
                           351 to 700            1.060            22
                           401 to 800            1.045            22
                           451 to 900            1.060            22
                           501 to 1000           1.045            22
                           551 to 1100           1.046            22
                           601 to 1200           1.051            22
                           651 to 1300           1.049            22
                           701 to 1400           1.044            22
                           751 to 1500           1.032            22
                           801 to 1600           1.037            22
                           851 to 1700           1.038            22
                           901 to 1800           1.031            22
                           951 to 1900           1.024            22
                          1001 to 2000           1.022            22
```

The next output excerpt shows information about the data and the variables in the imputation models.

```
DATA INFORMATION:

  Sample Size:              138
  Missing Data Rates:

                 behsymp1 = 03.62
                 lrnprob1 = 02.17
                    read1 = 06.52
                    read2 = 09.42
                    read9 = 17.39
                stanread7 = 19.57

VARIABLES IN IMPUTATION MODEL:

  Incomplete continuous:  behsymp1 lrnprob1 read1 read2 read9 stanread7

  NUMBER OF PARAMETERS
  Imputation Models:      42
```

MCMC estimation produces a distribution for each parameter in every unique imputation model. The median and standard deviation columns describe the center and spread of the posterior distributions; although they make no reference to drawing repeated samples, they are analogous—and numerically equivalent in most cases—to frequentist point estimates and standard errors. The 95% credible intervals in the rightmost columns give a range that captures 95% of the parameter's distribution. These are akin to confidence intervals, but the intervals describe parameter distributions rather than characteristics of repeated samples. The `N_Eff` values in rightmost column of the table give the effective number of MCMC samples for each parameter. These quantities essentially represent the number of independent estimates on which the parameter summaries are based after removing autocorrelations from the MCMC process. Gelman et al. (2014, p. 287) recommend values greater than 100. All values in the example table exceed this recommended minimum. In cases where the `N_Eff` values are insufficient, increasing the value on the `ITERATIONS` command will remedy the issue.

The Blimp output includes tables of regression parameters for every incomplete variable's imputation model. The imputation model parameters are not of substantive interest and would not be reported. An example table is shown below.

```
Missing variable:   behsymp1

Parameters                          Median     StdDev       2.5%      97.5%        PSR      N_Eff
                                ------------------------------------------------------------------

Grand Mean                          49.506      1.093     47.355     51.590      1.006   2075.694

Level 1:
  lrnprob1                           0.731      0.071      0.591      0.872      1.002   8316.690
  read1                             -0.274      0.077     -0.422     -0.121      1.002   8393.286
  read2                              0.590      0.103      0.386      0.792      1.002   7435.432
  read9                             -0.457      0.102     -0.657     -0.254      1.003   8587.332
  stanread7                         -0.018      0.016     -0.048      0.014      1.003   6881.275
  Residual Var.                     55.104      7.569     42.773     72.497      1.003   7269.684

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

When saving imputations, the bottom of the Blimp output file displays a table listing the order of the variables in the output data sets. All variables are saved regardless of whether they appeared in the fitted models. When saving data to a stacked file (e.g., for analysis in R or other packages), the first variable in the file is an integer index that identifies which data set each row belongs to (e.g., an integer variable that ranges from 1 to 20 in this example).

```
VARIABLE ORDER IN IMPUTED DATA:

separate = './imps/imp*.dat'

   id male hispanic riskgrp atrisk behsymp1 lrnprob1 read1 read2 read3
   read9 read9grp stanread7 math1 math2 math3 math9 math9grp stanmath7

stacked = './imps/imps.dat'

    imp# id male hispanic riskgrp atrisk behsymp1 lrnprob1 read1 read2 read3
    read9 read9grp stanread7 math1 math2 math3 math9 math9grp stanmath7
```

The imputed data sets are subsequently analyzed in another software package, and estimates and standard errors are combined using Rubin's rules (Little & Rubin, 2020). The analysis phase does not utilize the auxiliary variables, as their information is embedded in the imputations. Scripts for analyzing the imputed data sets are found in the next subsections.

Analyzing Imputations in Mplus

In lieu of the Bayesian estimates, Blimp's `SAVE` command can be used to save multiple imputations for analysis in the frequentist framework. Returning to the previous Blimp script, the `SAVE` command and the `separate` keyword saved each imputed data set to a separate file with the asterisk replaced by a numeric index. The separate keyword also creates a list of file names needed for analysis in Mplus (in this example, a filed called `implist.dat` located in the `imps` subfolder). The contents of this file were shown in Example 12.

The Mplus input file for analyzing the imputations is `Ex15.inp`. The script is virtually identical to the `Ex6.1.inp` file described in Example 1 with three exceptions. First, instead of naming the raw data set, the `DATA` command lists the text file containing the names of the imputed data sets (the `implist.dat` file located in the `./imps` subdirectory). The `type = imputation` subcommand instructs Mplus that the input data is a list of file names. Second, the missing subcommand is omitted because the analysis variables are now complete. Finally, the `MODEL` section no longer specifies a normal distribution for the predictors. Readers can refer back to Example 1 for a detailed description of the other commands. The code block below shows the analysis and pooling script.

Mplus Script Ex13.inp

```
DATA:
file = ./imps/implist.dat;
type = imputation;
VARIABLE:
names = id male hispanic riskgrp atrisk behsymp1 lrnprob1
   read1 read2 read3 read9 read9grp stanread7
   math1 math2 math3 math9 math9grp stanmath7;
usevariables = read9 read1 lrnprob1 behsymp1;
MODEL:
read9 on read1 lrnprob1 behsymp1 (beta1-beta3);
MODEL TEST:
0 = beta1; 0 = beta2; 0 = beta3;
OUTPUT:
stdyx cinterval;
```

Mplus Output

When fitting regression models to complete data sets, researchers often use an omnibus *F* test to evaluate the set of slope coefficients. The `MODEL TEST` command specified a multiple imputation Wald chi-square statistic evaluating the null hypothesis that the population slopes equal 0 (Asparouhov & Muthén, 2010b). The chi-square statistic, degrees of freedom, and *p*-value appear near the bottom of the `MODEL FIT INFORMATION` section under the `Wald Test of Parameter Constraints` heading. The test statistic is statistically significant, thus refuting the null hypothesis.

```
MODEL FIT INFORMATION

Number of Free Parameters                        5

...

Wald Test of Parameter Constraints

          Value                            175.893
          Degrees of Freedom                     3
          P-Value                           0.0000
```

The table of unstandardized parameter estimates is shown below. The first two columns display the pooled unstandardized estimates and standard errors, and the third and fourth columns display the corresponding *z*-statistics and *p*-values. The focal model results are shown in bold typeface. The `Rate of Missing` column (also called the fraction of missing information in the literature) quantifies the imputation noise in each estimate as proportion of its squared standard error.

```
MODEL RESULTS

                                                    Two-Tailed   Rate of
                  Estimate       S.E.  Est./S.E.    P-Value    Missing

 READ9    ON
    READ1            0.506      0.043     11.868      0.000      0.182
    LRNPROB1        -0.231      0.113     -2.047      0.041      0.149
    BEHSYMP1        -0.189      0.101     -1.864      0.062      0.160

 Intercepts
    READ9           65.487      5.803     11.284      0.000      0.150

 Residual Variances
    READ9           86.366     11.202      7.710      0.000      0.138
```

The results are interpreted in the same way as a complete-data regression analysis. For example, consider the first-grade reading score slope. The model predicts that two individuals who differ by one point on `READ1` but are the same on `LRNPROB1` and `BEHSYMP1` should differ by .51 points on `READ9`. The corresponding test statistic indicates that the slope coefficient is statistically different from zero ($z = 11.87$, $p < .001$). Note that these estimates are numerically identical to those from Bayesian and maximum likelihood estimation.

Specifying the `stdyx` keyword with the `OPTIONS` command prints the table of standardized estimates and R-squared statistics shown below. The slope coefficients convey the expected change in standard deviation units for a one standard deviation increase in a given predictor. For example, the model predicts that two individuals who differ by one standard deviation on `READ1` but are the same on `LRNPROB1` and `BEHSYMP1` should differ by .70 standard deviations on `READ9`. Collectively, the predictors explain 61% of the variation in ninth-grade reading scores.

```
STANDARDIZED MODEL RESULTS

STDYX Standardization

                                                    Two-Tailed   Rate of
                  Estimate       S.E.  Est./S.E.    P-Value    Missing

READ9    ON
   READ1             0.701      0.044     15.767      0.000      0.102
   LRNPROB1         -0.168      0.082     -2.036      0.042      0.157
   BEHSYMP1         -0.153      0.082     -1.861      0.063      0.159

Intercepts
   READ9             4.424      0.531      8.332      0.000      0.152

Residual Variances
   READ9             0.394      0.055      7.166      0.000      0.099
```

```
R-SQUARE

    Observed                                          Two-Tailed   Rate of
    Variable        Estimate       S.E.  Est./S.E.    P-Value    Missing

    READ9              0.606      0.055     11.033      0.000      0.099
```

Analyzing Imputations in R

Returning to the previous Blimp script, the `SAVE` command and the `stacked` keyword saved the imputed data sets to a single stacked file with an index variable in the first column identifying the individual files. The stacked file is appropriate for analyzing data in R, SAS, SPSS, and Stata, among others.

The R input file for the analysis is `Ex15.R`. The code block below shows the commands that import the data.

R Script Ex13.R

```
1   library(fdir)
2   library(mitml)
3   set()
4   imps <- read.table("./imps/imps.dat")
5   names(imps) <- c("imputation","id","male","hispanic","riskgrp",
6     "atrisk","behsymp1","lrnprob1","read1","read2","read3",
7     "read9","read9grp","stanread7","math1","math2","math3",
8     "math9","math9grp","stanmath7")
```

The example requires the `fdir` and `lavaan` packages, which are loaded on lines 1 and 2. On line 3, the `set()` function of the `fdir` package identifies the file path of the folder containing the R script and sets this location as the working directory. On line 4, the `read.table` command imports the stacked data. It is only necessary to specify the name of the input data file. No file path is required when the data reside in the same folder as the R script as is the case here. Finally, variable names are listed beginning on line 5. Importantly, the first variable named `IMPUTATION` is the index that identifies the individual files.

The next block of code relies on the `mitml` package to fit the model to each data set and pool the results using Rubin's rules. The `implist` command on line 9 unstacks the data and creates a list that contains the individual files. Line 10 fits the focal regression model using the `lm` function, and line 11 uses the `testEstimates` function in `mitml` to implement Rubin's pooling rules and save the results in an object called `estimates`. The `df.com` parameter is the denominator degrees of freedom that would have resulted had there been no missing data (i.e., $N–K–1$ degrees of freedom, where K is the number of predictors). This argument produces Barnard and Rubin degrees of freedom values. Finally, lines 12 and 13 print the estimates and confidence intervals.

R Script Ex13.R, continued

```
 9   implist <- as.mitml.list(split(imps, imps$imputation))
10   fit <- with(implist, lm(read9 ~ read1 + lrnprob1 + behsymp1))
11   estimates <- testEstimates(fit, extra.pars = T, df.com = 134)
12   estimates
13   confint(estimates)
```

When fitting regression models to complete data sets, researchers often use an omnibus F test to evaluate the set of slope coefficients. The `testModels` command below specifies a multiple imputation Wald F statistic evaluating the null hypothesis that the population slopes equal 0 (Li et al., 1991). The test requires an additional model on line 14 that represents the null hypothesis, which in this case is an empty regression model with just an intercept. On line 15, the full model and null model objects passed into the `testModels` function, and the `D1` keyword requests the Wald test. As before, the `df.com` parameter is the denominator degrees of freedom that would have resulted had there been no missing data. This argument produces the Barnard and Rubin (1999) degrees of freedom adjustment.

R Script Ex13.R, continued

```
14   null <- with(implist, lm(read9 ~ 1))
15   testModels(fit, null, df.com = 134, method = "D1")
```

R Output

The table of unstandardized pooled parameter estimates is shown below. The first two columns display the pooled unstandardized estimates and standard errors, and the third through fifth columns display the corresponding test statistics. The focal model results are shown in bold typeface. The `RIV` column (relative increase in variance) is a fraction comparing imputation noise to complete-data sampling variation, and the `FMI` column (fraction of missing information in the literature) quantifies the imputation noise in each estimate as proportion of its squared standard error.

```
> estimates

Call:

testEstimates(model = fit, extra.pars = T, df.com = 134)

Final parameter estimates and inferences obtained from 20 imputed data sets.

             Estimate Std.Error   t.value        df   P(>|t|)       RIV       FMI
(Intercept)    65.487     5.877    11.144   100.498     0.000     0.169     0.161
read1           0.506     0.043    11.725    92.752     0.000     0.212     0.192
lrnprob1       -0.231     0.114    -2.022   100.704     0.046     0.168     0.160
behsymp1       -0.189     0.102    -1.841    97.962     0.069     0.182     0.171

                   Estimate
Residual~~Residual   88.944

Hypothesis test adjusted for small samples with df=[134]
complete-data degrees of freedom.

> confint(estimates)
                  2.5 %      97.5 %
(Intercept) 53.8288728 77.14584684
read1        0.4202903  0.59168880
lrnprob1    -0.4581615 -0.00433096
behsymp1    -0.3919669  0.01475078
```

The results are interpreted in the same way as a complete-data regression analysis. For example, consider the first-grade reading score slope. The model predicts that two individuals who differ by one point on `READ1` but are the same on `LRNPROB1` and `BEHSYMP1` should differ by .51 points on `READ9`. The corresponding test statistic indicates that the slope coefficient is statistically different from zero ($t = 11.73$, $p < .001$). Note that these estimates are numerically identical to those from Bayesian and maximum likelihood estimation.

Finally, the Wald omnibus F statistic is shown in the output table below. The test statistic is statistically significant, thus refuting the null hypothesis that all population slopes equal zero.

```
Model comparison calculated from 20 imputed data sets.
Combination method: D1

    F.value     df1     df2   P(>F)     RIV
     58.272       3 123.487   0.000   0.177

Hypothesis test adjusted for small samples with df=[134]
complete-data degrees of freedom.
```

EXAMPLE 14: FULLY CONDITIONAL SPECIFICATION IMPUTATION WITH MIXED VARIABLE TYPES

This example illustrates model-agnostic fully conditional specification multiple imputation with mixed variable types. The analysis uses the `behaviorachievement.dat` data set taken from a longitudinal study that followed 138 students from primary through middle school. The file includes three annual assessments of broad reading and math achievement beginning in the first grade, seventh grade standardized achievement test scores taken from a statewide assessment, and a final measure of broad reading and math obtained in ninth grade. The data also contain teacher ratings of behavioral symptoms and learning problems were also obtained in the first grade. The data description at the beginning of this document provides additional details. The variables for this analysis are as follows.

Name	Definition	Missing %	Scale
	Focal Variables		
RISKGRP	Emotional/behavioral disorder risk	2.2	1 = Low, 2 = Medium, 3 = High
BEHSYMP1	1st grade behavioral symptoms	3.6	Numeric
LRNPROB1	1st grade learning problems	2.2	Numeric
READ1	1st grade broad reading composite	6.5	Numeric
READ9	9th grade broad reading composite	17.4	Numeric
	Auxiliary Variables		
READ2	2nd grade broad reading composite	9.4	Numeric
STANREAD7	7th grade standardized math	19.6	Numeric

Imputation and Analysis Models

Fully conditional specification uses a sequence of regression models to fill in missing values. Specifically, each MCMC iteration fits a series of models where one incomplete variable is regressed on all other variables. The predicted values and residual variance from each model define the center and spread of the imputed values, which are drawn at random from a normal distribution. After imputing the missing scores, the filled-in variable becomes a predictor in all other imputation models in the sequence. The imputation stage should include all variables and effects for the subsequent analyses, and it should incorporate auxiliary variables that either predict missingness or correlate with the incomplete variables (Collins et al., 2001).

A common goal of model-agnostic imputation is to generate imputations for different purposes (e.g., descriptive summaries, several analyses within the same project). To illustrate an entire multiple imputation analysis, suppose that one use of the filled-in data sets involves a model where ninth grade broad reading scores are regressed on first grade reading achievement, teacher-rated learning problems and behavioral symptoms, and a three-category nominal variable indicating risk for emotional or behavioral disorders.

$$\begin{aligned} READ_9 = \beta_0 + \beta_1(READ_1) + \beta_2(LRNPROB_1) + \beta_3(BEHSYMP_1^*) \\ + \beta_4(MEDRISK) + \beta_5(HIGHRISK) + \varepsilon \end{aligned} \tag{28}$$

The `MEDRISK` and `HIGHRISK` variables are dummy code variables that contrast the medium- and high-risk groups, respectively, against the low-risk reference group. Example 9 used the same analysis model to illustrate Bayesian estimation and model-based multiple imputation.

Blimp Script

The code block below shows Blimp script `Ex14.inp`.

Blimp Script Ex14.imp

```
DATA: behaviorachievement.dat;
VARIABLES: id male hispanic riskgrp atrisk behsymp1 lrnprob1
  read1 read2 read3 read9 read9grp stanread7
  math1 math2 math3 math9 math9grp stanmath7;
NOMINAL: riskgrp;
MISSING: 999;
FCS: read9 read1 lrnprob1 behsymp1 riskgrp stanread7 read2;
SEED: 90291;
BURN: 1000;
ITERATIONS: 10000;
NIMPS: 20;
CHAINS: 20;
SAVE:
stacked = ./imps/imps.dat;
separate = ./imps/imp*.dat;
```

The first six lines can be viewed as a set of commands that specify information about the data and variables. The `DATA` command specifies the name of the input text file. No file path is required when the data file is located in the same directory as the script, as it is here. Starting on line 2, the `VARIABLES` command names the data columns. The `NOMINAL` command on line 5 identifies the multicategorical nominal predictor, and the `MISSING` command on line 6 defines a global missing value code as 999.

Next, the `FCS` command lists all variables—complete or incomplete—that are included in the imputation phase. Using the `FIXED` command to identify complete variables reduces computational time because these variables do not require a regression model (see Example 12). Lines 8 through 10 can also be viewed as a block of commands that specify features of the MCMC algorithm: the `SEED` command gives an integer string that initializes the random number generator, the `BURN` command specifies the number of iterations for the warm-up or burn-in period, and the `ITERATIONS` command gives the number of MCMC iterations on which the imputation model summaries are based (essentially, the total number of MCMC cycles across all chains following the warm-up period).

The `NIMPS`, `CHAINS` and `SAVE` commands can be viewed as a set. Setting `NIMPS` equal to `CHAINS` saves a single filled-in data set from the final iteration of a unique `MCMC` process, thus avoiding autocorrelation among the imputations. The `SAVE` command provides a name for the imputed data sets. The script illustrates how to save data sets in two common formats. The stacked keyword creates a stacked file where all imputations are in a single file, and the separate keyword saves each imputed data set to a separate file with the asterisk replaced by a numeric index. To keep things organized, the `./imps` part of the file path points to a subfolder named `imps` located within the same folder as the script and data. The separate keyword also creates a list of file names needed for analysis in Mplus (in this example, a filed called `implist.dat` located in the `imps` folder).

Blimp Output

Prior to inspecting the parameter estimates, it is important to investigate the potential scale reduction (PSR) factor diagnostics (Gelman & Rubin, 1992) to determine whether MCMC has converged. Blimp divides the burn-in period into 20 equal segments, and it computes the PSR diagnostic for every parameter. The table located near the top of the output reports the highest (worst) PSR value across all parameters in every model. A common recommendation is that these values should be less than 1.05 or perhaps 1.10 (Asparouhov & Muthén, 2010a; Gelman et

al., 2014). If the PSR in the bottom row of the table (the final check of the burn-in period) is above these cutoffs, then rerun the analysis with a longer burn-in period.

```
BURN-IN POTENTIAL SCALE REDUCTION (PSR) OUTPUT:

  NOTE: Split chain PSR is being used. This splits each chain's
        iterations to create twice as many chains.

  Comparing iterations across 2 chains     Highest PSR   Parameter #
                           51 to 100             1.417            45
                          101 to 200             1.156            45
                          151 to 300             1.243            45
                                 ...               ...            ..
                          901 to 1800            1.022            54
                          951 to 1900            1.021            45
                         1001 to 2000            1.021            45
```

The next output excerpt shows information about the data and the variables in the imputation models.

```
DATA INFORMATION:

  Sample Size:              138
  Missing Data Rates:

                  riskgrp = 02.17
                 behsymp1 = 03.62
                 lrnprob1 = 02.17
                    read1 = 06.52
                    read2 = 09.42
                    read9 = 17.39
                stanread7 = 19.57
```

```
Nominal Dummy Codes:

                  riskgrp = riskgrp.2 riskgrp.3


VARIABLES IN IMPUTATION MODEL:

  Incomplete continuous:  behsymp1 lrnprob1 read1 read2 read9 stanread7
  Incomplete nominal:     riskgrp

  NUMBER OF PARAMETERS
 Imputation Models:      68
```

MCMC estimation produces a distribution for each parameter in every unique imputation model. The median and standard deviation columns describe the center and spread of the posterior distributions; although they make no reference to drawing repeated samples, they are analogous—and numerically equivalent in most cases—to frequentist point estimates and standard errors. The 95% credible intervals in the rightmost columns give a range that captures 95% of the parameter's distribution. These are akin to confidence intervals, but the intervals describe parameter distributions rather than characteristics of repeated samples. The `N_Eff` values in rightmost column of the table give the effective number of MCMC samples for each parameter. These quantities essentially represent the number of independent estimates on which the parameter summaries are based after removing autocorrelations from the MCMC process. Gelman et al. (2014, p. 287) recommend values greater than 100. All values in the example table exceed this recommended minimum. In cases where the `N_Eff` values are insufficient, increasing the value on the `ITERATIONS` command will remedy the issue.

The Blimp output includes tables of regression parameters for every incomplete variable's imputation model. The imputation model parameters are not of substantive interest and would not be reported. An example table is shown below.

```
Missing variable:   behsymp1

Parameters                          Median     StdDev       2.5%      97.5%        PSR      N_Eff
                                ------------------------------------------------------------------

Grand Mean                          49.596      1.101     47.401     51.765      1.007   2184.881

Level 1:
  riskgrp.2                         -0.581      1.093     -2.657      1.578      1.006   3551.667
  riskgrp.3                          1.574      1.268     -1.031      3.876      1.009   1864.907
  lrnprob1                           0.705      0.079      0.547      0.856      1.005   4786.167
  read1                             -0.217      0.093     -0.393     -0.032      1.004   3349.964
  read2                              0.593      0.108      0.381      0.807      1.004   5270.399
  read9                             -0.447      0.105     -0.652     -0.242      1.004   5961.720
  stanread7                         -0.016      0.016     -0.049      0.016      1.003   5148.240
  Residual Var.                     52.691      7.826     39.338     70.152      1.005   4634.464

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

When saving imputations, the bottom of the Blimp output file displays a table listing the order of the variables in the output data sets. All variables are saved regardless of whether they appeared in the fitted models. When saving data to a stacked file (e.g., for analysis in R or other packages), the first variable in the file is an integer index that identifies which data set each row belongs to (e.g., an integer variable that ranges from 1 to 20 in this example).

```
VARIABLE ORDER IN IMPUTED DATA:

separate = './imps/imp*.dat'

   id male hispanic riskgrp atrisk behsymp1 lrnprob1 read1 read2 read3
  read9 read9grp stanread7 math1 math2 math3 math9 math9grp stanmath7

stacked = './imps/imps.dat'

   imp# id male hispanic riskgrp atrisk behsymp1 lrnprob1 read1 read2 read3
   read9 read9grp stanread7 math1 math2 math3 math9 math9grp stanmath7
```

The imputed data sets are subsequently analyzed in another software package, and estimates and standard errors are combined using Rubin's rules (Little & Rubin, 2020). The analysis phase does not utilize the auxiliary variables, as their information is embedded in the imputations. Scripts for analyzing the imputed data sets are found in the next subsections.

Analyzing Imputations in Mplus

In lieu of the Bayesian estimates, Blimp's `SAVE` command can be used to save multiple imputations for analysis in the frequentist framework. Returning to the previous Blimp script, the `SAVE` command and the `separate` keyword saved each imputed data set to a separate file with the asterisk replaced by a numeric index. The separate keyword also creates a list of file names needed for analysis in Mplus (in this example, a filed called `implist.dat` located in the `imps` subfolder). Example 6 shows the contents of this file.

The Mplus input file for analyzing the imputations is `Ex16.inp`. The script is similar to previous Mplus scripts (e.g., the `Ex1.1.inp` file described in Example 1) with four exceptions. First, instead of naming the raw data set, the `DATA` command lists the text file containing the names of the imputed data sets (the `implist.dat` file located in the `./imps` subdirectory). The `type = imputation` subcommand instructs Mplus that the input data is a list of file names. Second, the missing subcommand is omitted because the analysis variables are now complete. Third, the `MODEL` section no longer specifies a normal distribution for the predictors or models for the auxiliary variables. Finally, lines 9 through 13 use the `DEFINE` command to create a pair of dummy codes. Lines 10 and 11 initialize a pair of new variables (`RISKGRP2` and `RISKGRP3`) with all 0s, and lines 12 and 13 recode these variables into dummy variables. Importantly, new variables computed with the `DEFINE` command must appear at the end of the `usevariables` list on line 8. The code block below shows the analysis and pooling script.

Mplus Script Ex14.inp

```
1   DATA:
2   file = ./imps/implist.dat;
3   type = imputation;
4   VARIABLE:
5   names = id male hispanic riskgrp atrisk behsymp1 lrnprob1
6     read1 read2 read3 read9 read9grp stanread7
7     math1 math2 math3 math9 math9grp stanmath7;
8   usevariables = read9 read1 lrnprob1 behsymp1 riskgrp2 riskgrp3;
9   DEFINE:
10  riskgrp2 = 0;
11  riskgrp3 = 0;
12  if(riskgrp eq 2) then riskgrp2 = 1;
13  if(riskgrp eq 3) then riskgrp3 = 1;
```

```
MODEL:
read9 on read1 lrnprob1 behsymp1 riskgrp2 riskgrp3 (beta1-beta5);
MODEL TEST:
0 = beta1; 0 = beta2; 0 = beta3;
OUTPUT:
stdyx cinterval;
```

Mplus Output

When fitting regression models to complete data sets, researchers often use an omnibus *F* test to evaluate the set of slope coefficients. The `MODEL TEST` command specified a multiple imputation Wald chi-square statistic evaluating the null hypothesis that the population slopes equal 0 (Asparouhov & Muthén, 2010b). The chi-square statistic, degrees of freedom, and *p*-value appear near the bottom of the `MODEL FIT INFORMATION` section under the `Wald Test of Parameter Constraints` heading. The test statistic is statistically significant, thus refuting the null hypothesis.

```
MODEL FIT INFORMATION

Number of Free Parameters                       7

...

Wald Test of Parameter Constraints

          Value                            173.432
          Degrees of Freedom                     5
          P-Value                           0.0000
```

The table of unstandardized parameter estimates is shown below. The first two columns display the pooled unstandardized estimates and standard errors, and the third and fourth columns display the corresponding *z*-statistics and *p*-values. The focal model results are shown in bold typeface. The `Rate of Missing` column (also called the fraction of missing information in

the literature) quantifies the imputation noise in each estimate as proportion of its squared standard error.

```
MODEL RESULTS

                                                    Two-Tailed   Rate of
                  Estimate       S.E.  Est./S.E.    P-Value    Missing

 READ9    ON
    READ1            0.477      0.047     10.122      0.000      0.134
    LRNPROB1        -0.250      0.115     -2.173      0.030      0.172
    BEHSYMP1        -0.166      0.106     -1.566      0.117      0.228
    RISKGRP2        -1.710      1.882     -0.908      0.364      0.076
    RISKGRP3        -3.115      2.820     -1.105      0.269      0.272

 Intercepts
    READ9           69.174      6.218     11.125      0.000      0.154

 Residual Variances
    READ9           85.516     11.867      7.206      0.000      0.249
```

The results are interpreted in the same way as a complete-data regression analysis. For example, consider the first-grade reading score slope. The model predicts that two individuals who differ by one point on `READ1` but are the same on all other predictors should differ by 0.48 points on `READ9`. The corresponding test statistic indicates that the slope coefficient is statistically different from zero ($z = 10.29$, $p < .001$). The two dummy codes appear as `RISKGRP2` and `RISKGRP3`. Consistent with a complete-data regression analysis, the dummy code slopes represent mean differences relative to the low-risk reference group. For example, holding all other predictors constant, the model predicts that a high-risk study would score 3.12 points lower than a low-risk student in the comparison group. Note that these estimates are virtually identical to those from Bayesian estimation. The output also includes a table with standardized coefficients and the *R*-squared statistic.

Analyzing Imputations in R

Returning to the previous Blimp script, the `SAVE` command and the `stacked` keyword saved the imputed data sets to a single stacked file with an index variable in the first column identifying the

individual files. The stacked file is appropriate for analyzing data in R, SAS, SPSS, and Stata, among others.

The R input file for the analysis is `Ex16.R`. The code block below shows the commands that import the data.

R Script Ex14.R

```
1   library(fdir)
2   library(mitml)
3   set()
4   imps <- read.table("./imps/imps.dat")
5   names(imps) <- c("imputation","id","male","hispanic","riskgrp",
6     "atrisk","behsymp1","lrnprob1","read1","read2","read3",
7     "read9","read9grp","stanread7","math1","math2","math3",
8     "math9","math9grp","stanmath7")
9   imps$riskgrp <- factor(imps$riskgrp)
```

The example requires the `fdir` and `lavaan` packages, which are loaded on lines 1 and 2. On line 3, the `set()` function of the `fdir` package identifies the file path of the folder containing the R script and sets this location as the working directory. On line 4, the `read.table` command imports the stacked data. It is only necessary to specify the name of the input data file. No file path is required when the data reside in the same folder as the R script as is the case here. Variable names are listed beginning on line 5. Importantly, the first variable named `IMPUTATION` is the index that identifies the individual files. Finally, line 9 defines the `RISKGRP` variable as a factor with qualitatively different levels. This specification will automatically introduce a set of dummy codes into the regression model.

The next block of code relies on the `mitml` package to fit the model to each data set and pool the results using Rubin's rules.

R Script Ex14.R, continued

```
10  implist <- as.mitml.list(split(imps, imps$imputation))
11  fit <- with(implist, lm(read9 ~ read1 + lrnprob1 + behsymp1 + riskgrp))
12  estimates <- testEstimates(fit, extra.pars = T, df.com = 132)
13  estimates
14  confint(estimates)
```

The `implist` command on line 10 unstacks the data and creates a list that contains the individual files. Line 11 fits the focal regression model using the `lm` function, and line 12 uses the `testEstimates` function in `mitml` to implement Rubin's pooling rules and save the results in an object called `estimates`. The `df.com` parameter is the denominator degrees of freedom that would have resulted had there been no missing data (i.e., $N–K–1$ degrees of freedom, where K is the number of predictors). This argument produces Barnard and Rubin degrees of freedom values. Finally, lines 13 and 14 print the estimates and confidence intervals.

When fitting regression models to complete data sets, researchers often use an omnibus F test to evaluate the set of slope coefficients. The `testModels` command below specifies a multiple imputation Wald F statistic evaluating the null hypothesis that the population slopes equal 0 (Li et al., 1991). The test requires an additional model on line 15 that represents the null hypothesis, which in this case is an empty regression model with just an intercept. On line 16, the full model and null model objects passed into the `testModels` function, and the `D1` keyword requests the Wald test. As before, the `df.com` parameter is the denominator degrees of freedom that would have resulted had there been no missing data. This argument produces the Barnard and Rubin (1999) degrees of freedom adjustment.

R Script Ex14.R, continued

```
15  null <- with(implist, lm(read9 ~ 1))
16  testModels(fit, null, df.com = 132, method = "D1")
```

R Output

The table of unstandardized pooled parameter estimates is shown below. The first two columns display the pooled unstandardized estimates and standard errors, and the third through fifth

columns display the corresponding test statistics. The focal model results are shown in bold typeface. The `RIV` column (relative increase in variance) is a fraction comparing imputation noise to complete-data sampling variation, and the `FMI` column (fraction of missing information in the literature) quantifies the imputation noise in each estimate as proportion of its squared standard error.

```
> estimates

Call:

testEstimates(model = fit, extra.pars = T, df.com = 132)

Final parameter estimates and inferences obtained from 20 imputed data sets.

             Estimate Std.Error   t.value        df   P(>|t|)       RIV       FMI
(Intercept)    69.174     6.337    10.916    98.577     0.000     0.172     0.164
read1           0.477     0.048     9.928   103.392     0.000     0.146     0.144
lrnprob1       -0.250     0.117    -2.133    94.276     0.036     0.196     0.181
behsymp1       -0.166     0.108    -1.539    81.473     0.128     0.276     0.235
riskgrp2       -1.710     1.921    -0.890   116.647     0.375     0.079     0.088
riskgrp3       -3.115     2.867    -1.087    72.059     0.281     0.348     0.278

                   Estimate
Residual~~Residual   89.403

Hypothesis test adjusted for small samples with df=[132]
complete-data degrees of freedom.

> confint(estimates)
                  2.5 %      97.5 %
(Intercept) 56.5999199 81.74779632
read1        0.3820806  0.57283035
lrnprob1    -0.4822008 -0.01730107
behsymp1    -0.3796984  0.04849960
riskgrp2    -5.5147538  2.09562507
riskgrp3    -8.8300720  2.59967016
```

The results are interpreted in the same way as a complete-data regression analysis. For example, consider the first-grade reading score slope. The model predicts that two individuals who differ by one point on `READ1` but are the same on all other predictors should differ by 0.48 points on `READ9`. The corresponding test statistic indicates that the slope coefficient is statistically different from zero ($t = 9.93$, $p < .001$). The two dummy codes appear as `RISKGRP2` and `RISKGRP3`. Consistent with a complete-data regression analysis, the dummy code slopes represent mean differences relative to the low-risk reference group. For example, holding all other predictors constant, the model predicts that a high-risk study would score 3.12 points

lower than a low-risk student in the comparison group. Note that these estimates are virtually identical to those from Bayesian and maximum likelihood estimation.

Finally, the Wald omnibus *F* statistic is shown in the output table below. The test statistic is statistically significant, thus refuting the null hypothesis that all population slopes equal zero.

```
Model comparison calculated from 20 imputed data sets.
Combination method: D1

    F.value     df1     df2   P(>F)     RIV
     33.252       5 123.203   0.000   0.213

Hypothesis test adjusted for small samples with df=[132]
complete-data degrees of freedom.
```

SECTION 4: MULTILEVEL MODELS

EXAMPLE 15: TWO-LEVEL REGRESSION WITH RANDOM INTERCEPTS

This example illustrates a two-level multiple regression with random intercepts. The analysis uses the `problemsolving2level.dat` data set taken from a cluster-randomized educational intervention where 29 schools were assigned to an intervention and comparison condition. In addition to the intervention assignment indicator, school-level variables include the average years of teacher experience and the percentage of learners for whom English is a second language. The 928 student-level records include pretest and posttest math problem-solving and self-efficacy scores, standardized math scores taken from a statewide assessment, and several sociodemographic variables. The analysis variables are as follows.

Name	Definition	Missing %	Scale
	Identifier Variables		
SCHOOL	School identifier	0	Integer index
	Focal Variables		
PSOLVEPRE	Math problem-solving pretest	0	Numeric
MALE	Gender dummy code	0	0 = Female, 1 = Male
FRLUNCH	Lunch assistance code	4.7	0 = None, 1 = Free/reduced lunch
TEACHEXP	Teacher years of experience	10.3	Numeric
CONDITION	Experimental condition	0	0 = Control, 1 = Experimental
	Auxiliary Variables		
LOWACH	Low achievement code	2.1	0 = Typical, 1 = Low achieving
STANMATH	Standardized math scores	7.4	Numeric

Analysis Model

The analysis is a random intercept regression model featuring problem-solving posttest scores regressed on the experimental condition dummy code at level-2 and four covariates, all of which are grand mean centered: problem-solving pretest scores (level-1), gender and lunch assistance dummy codes (level-1), and years of teacher experience (level-2). To convey each variable's level, the *i* and *j* subscripts denote students and schools, respectively.

$$PSOLVEPST_{ij} = (\beta_0 + b_{0j}) + \beta_1(PSOLVEPRE_{ij}) + \beta_2(MALE_{ij}) + \beta_3(FRLUNCH_{ij}) + \beta_4(TEACHEXP_j) + \beta_5(CONDITION_j) + \varepsilon_{ij} \quad (29)$$

Unlike a complete-data regression analysis, all incomplete variables require distributional assumptions, including the predictors. Blimp uses a factored regression specification that assigns separate distributions to the predictors and outcome. By default, Blimp invokes a multivariate normal distribution for numeric predictors and the latent response scores for discrete predictors.

The missing data literature often recommends an inclusive strategy that incorporates auxiliary variables that either predict missingness or correlate with the incomplete variables (Collins et al., 2001). Following earlier examples, auxiliary variables enter the model as additional level-1 outcomes that are predicted by the analysis variables and by each other. The additional regression equations are as follows.

$$\begin{aligned} LOWACH^*_{ij} &= \gamma_{01} + \gamma_{11}(PSOLVEPST_{ij}) + \gamma_{21}(PSOLVEPRE_{ij}) + \gamma_{31}(MALE_{ij}) \\ &\quad + \gamma_{41}(FRLUNCH_{ij}) + \gamma_{51}(TEACHEXP_j) + \gamma_{61}(CONDITION_j) + \epsilon_{1ij} \\ STANMATH &= \gamma_{02} + \gamma_{12}(LOWACH_{ij}) + \gamma_{22}(PSOLVEPST_{ij}) + \gamma_{32}(PSOLVEPRE_{ij}) \\ &\quad + \gamma_{42}(MALE_{ij}) + \gamma_{52}(FRLUNCH_{ij}) + \gamma_{62}(TEACHEXP_j) + \gamma_{72}(CONDITION_j) + \epsilon_2 \end{aligned} \quad (30)$$

The `LOWACH` model is a probit regression, with the binary outcome model as a latent response variable (denoted by the asterisk superscript). Again, the entire collection of regressions can be viewed as a path model, where the focal regression is one part of a larger network. The key difference is that the path coefficients are just a tool for linking incomplete variables and do not represent a substantive theory.

Blimp Script

The code block below shows Blimp script `Ex15.1.inp`. The first six lines can be viewed as a set of commands that specify information about the data and variables. The `DATA` command specifies the name of the input text file. No file path is required when the data file is located in the same directory as the script, as it is here. Starting on line 2, the `VARIABLES` command names the data columns. The `CLUSTERID` command on line 4 lists the school-level identifier variable that indicates the clustering of the data records in schools. Including the `CLUSTERID` command automatically introduces random intercepts. The `ORDINAL` command on line 5 identifies binary

and ordinal variables. Binary variables can be defined as ordinal or nominal, as the statistical models are identical. The `MISSING` command on line 6 defines a global missing value code as 999.

Blimp Script Ex15.1.imp

```
1   DATA: problemsolving2level.dat;
2   VARIABLES: school student condition teachexp eslpct ethnic male
3     frlunch lowach stanmath efficacypre efficacypst psolvepre psolvepst;
4   CLUSTERID: school;
5   ORDINAL: condition male frlunch lowach;
6   MISSING: 999;
7   FIXED: condition male psolvepre;
8   CENTER: grandmean = psolvepre male frlunch teachexp;
9   MODEL:
10  focal.model:
11  psolvepst ~ psolvepre@b1 male@b2 frlunch@b3 teachexp@b4 condition@b5;
12  auxiliary.models:
13  stanmath lowach  ~ psolvepst psolvepre male frlunch teachexp condition;
14  TEST: b1:b4 = 0;
15  TEST: b5 = 0;
16  SEED: 90291;
17  BURN: 3000;
18  ITERATIONS: 10000;
```

The `FIXED`, `CENTER`, `MODEL`, and `TEST` blocks can be viewed as a set. The `FIXED` command identifies a complete predictor, which does not require a distribution or regression model. The `CENTER` command deviates the four covariates at their iteratively-estimated grand means. Beginning on line 9, the `MODEL` command lists the regression models, with outcome variables to the left of the tilde and predictors to the right. The code uses labels (`focal.model` and `auxiliary.models`) to order output tables, such that the focal model appears first followed by the auxiliary variable models. The focal model listed on line 11 assigns labels to the slope coefficients using the @ symbol. Blimp automatically configures the explanatory variable models under the assumption that the numeric variables and latent response scores (discrete predictors) are normally distributed. Line 13 is a syntax shortcut that produces the two auxiliary variable regression models in Equation 30; in the first model, `LOWACH` is regressed on the focal variables, and the second model features `STANMATH` regressed on `LOWACH` and the focal variables. The `TEST`

commands on lines 14 and 15 use the parameter labels to specify a pair of custom hypothesis test, one that evaluates the set of covariates, and another that evaluates the intervention effect. These commands produce the Bayesian Wald test (Asparouhov & Muthén, 2021), which is essentially a chi-square statistic that captures the discrepancy between the Bayesian point estimates (posterior means) and the hypothesized values of zero.

Finally, lines 16 through 18 can be viewed as a block of commands that specify features of the MCMC algorithm: the `SEED` command gives an integer string that initializes the random number generator, the `BURN` command specifies the number of iterations for the warm-up or burn-in period, and the `ITERATIONS` command gives the number of MCMC iterations on which the analysis summaries are based (essentially, the number of MCMC cycles following the warm-up period).

Blimp Output

Prior to inspecting the parameter estimates, it is important to investigate the potential scale reduction (PSR) factor diagnostics (Gelman & Rubin, 1992) to determine whether MCMC has converged. Blimp divides the burn-in period into 20 equal segments, and it computes the PSR diagnostic for every parameter. The table located near the top of the output reports the highest (worst) PSR value across all parameters in every model. A common recommendation is that these values should be less than 1.05 or perhaps 1.10 (Asparouhov & Muthén, 2010a; Gelman et al., 2014). If the PSR in the bottom row of the table (the final check of the burn-in period) is above these cutoffs, then rerun the analysis with a longer burn-in period.

```
BURN-IN POTENTIAL SCALE REDUCTION (PSR) OUTPUT:

  NOTE: Split chain PSR is being used. This splits each chain's
        iterations to create twice as many chains.

  Comparing iterations across 2 chains     Highest PSR   Parameter #
                            76 to 150            1.342            34
                           151 to 300            1.553            35
                                  ...              ...            ..
                          1351 to 2700           1.027             8
                          1426 to 2850           1.046            13
                          1501 to 3000           1.040             8
```

The next section of the output displays information about the variables in the analysis and the models used for estimation. This output table mimics the one from Example 6, but it additionally reports the number of observations at each level.

```
DATA INFORMATION:

  Level-2 identifier:       school
  Sample Size:              982
  Level-2 Clusters:         29
  Missing Data Rates:

                psolvepst = 20.47
                   lowach = 02.14
                 stanmath = 07.43
                 teachexp = 10.34
                  frlunch = 04.68
```

The pair of `TEST` commands in the previous script requested a Bayesian Wald chi-square statistic (Asparouhov & Muthén, 2021) that evaluates two different null hypotheses. The first test evaluates the hypothesis that all covariate slopes equal zero, and the second evaluates just the intervention group mean difference. The chi-square statistic, degrees of freedom, and *p*-value appear near the bottom of the `MODEL FIT` section under the `WALD TEST` heading. The latter test is shown in the output excerpt below.

```
MODEL FIT:

  ...

WALD TESTS (Asparouhov & Muthén, 2021)

...

Test #2

    Full:
      [1]  psolvepst ~ Intercept psolvepre@beta1 male@beta2
           frlunch@beta3 teachexp@beta4 condition@beta5

    Restricted:
      [1]  psolvepst ~ Intercept psolvepre@beta1 male@beta2 frlunch@beta3
           teachexp@beta4 condition@beta5

    Constraints in Restricted:
      [1]  b5 = 0


    Wald Statistic (Chi-Square)                 4.088
    Number of Parameters Tested (df)                1
    Probability                                 0.043
```

The table summarizing the focal regression model includes unstandardized coefficients, standardized slopes, and variance explained effect size estimates (Rights & Sterba, 2019). MCMC estimation produces a distribution for each parameter in the table. The median and standard deviation columns describe the center and spread of the posterior distributions; although they make no reference to drawing repeated samples, they are analogous—and numerically equivalent in most cases—to frequentist point estimates and standard errors. The 95% credible intervals in the rightmost columns give a range that captures 95% of the parameter's distribution. These are akin to confidence intervals, but the intervals describe parameter distributions rather than characteristics of repeated samples. The `N_Eff` values in rightmost column of the table give the effective number of MCMC samples for each parameter. These quantities essentially represent the number of independent estimates on which the parameter summaries are based after

removing autocorrelations from the MCMC process. Gelman et al. (2014, p. 287) recommend values greater than 100. All values in the example table exceed this recommended minimum. In cases where the `N_Eff` values are insufficient, increasing the value on the `ITERATIONS` command will remedy the issue. The table is shown below.

```
OUTCOME MODEL ESTIMATES:

  Summaries based on 10000 iterations using 2 chains.

  focal.model block:

Outcome Variable:  psolvepst

Grand Mean Centered: frlunch psolvepre teachexp


Parameters                          Median     StdDev       2.5%      97.5%        PSR      N_Eff
                                  ------------------------------------------------------------------
Variances:
  L2 : Var(Intercept)                5.054      1.998      2.643     10.369      1.001   2647.814
  Residual Var.                     20.651      1.062     18.768     22.907      1.000   6040.557

Coefficients:
  Intercept                         52.876      0.725     51.402     54.243      1.002    611.789
  psolvepre                          0.460      0.036      0.390      0.531      1.000   5377.366
  male                               0.172      0.332     -0.490      0.807      1.000   5929.056
  frlunch                           -0.719      0.455     -1.575      0.192      1.000   4325.570
  teachexp                           0.120      0.115     -0.102      0.354      1.003    494.712
  condition                          1.844      0.921      0.077      3.687      1.008    505.513

Standardized Coefficients:
  psolvepre                          0.394      0.030      0.332      0.452      1.000   2936.056
  male                               0.015      0.028     -0.042      0.069      1.000   5918.541
  frlunch                           -0.050      0.031     -0.109      0.013      1.000   4262.038
  teachexp                           0.087      0.080     -0.073      0.240      1.003    504.074
  condition                          0.157      0.075      0.007      0.300      1.008    501.389

Proportion Variance Explained
  by Coefficients                    0.217      0.037      0.150      0.293      1.005    816.605
  by Level-2 Random Intercepts       0.153      0.048      0.085      0.272      1.002   2524.209
  by Level-1 Residual Variation      0.626      0.046      0.522      0.703      1.000   1270.578
                                  ------------------------------------------------------------------
```

The results are interpreted in the same way as a complete-data multilevel analysis. The first section of the output table displays the variance estimates. The random intercept and within-cluster residual variances are denoted `L2:Var(Intercept)` and `Residual Var.`, respectively. Moving to the coefficients section, the primary focus is the β_5 coefficient, the value of indicates

that intervention schools scored 1.84 points higher than control schools, on average, controlling for student- and school-level covariates. Consistent with the previous significance test, the 95% credible interval limits suggest this effect is statistically different from zero because the null value is outside the interval. The bottom portion of the table displays Rights and Sterba (2019) *R*-squared effect size values. The fixed effects explain 22% of the total variation, and the random intercepts account for 15% of the variability.

The Blimp output also includes tables of regression model parameters for the auxiliary variables as well as the auto-generated models for incomplete predictors. These additionally results are not of substantive interest and would not be reported. The auxiliary variable models appear in `OUTCOME MODEL ESTIMATES` section with the focal results, and the auto-generated predictor models are displayed under the heading `PREDICTOR MODEL ESTIMATES`.

Saving Multiple Imputations

MCMC estimation imputes missing values at every iteration, such that the resulting Bayesian estimates average over thousands of plausible replacement scores (10,000 sets in this example). A subset of the imputations can be saved for reanalysis in the frequentist framework, if desired. The Blimp input file `Ex15.2.imp` is identical `Ex15.1.imp`, but it adds the following lines at the bottom of the script.

```
NIMPS: 20;
CHAINS: 20;
SAVE:
stacked = ./imps/imps.dat;
separate = ./imps/imp*.dat;
```

The `NIMPS`, `CHAINS` and `SAVE` commands can be viewed as a set. Setting `NIMPS` equal to `CHAINS` saves a single filled-in data set from the final iteration of a unique `MCMC` process, thus avoiding autocorrelation among the imputations. The `SAVE` command provides a name for the imputed data sets. The script illustrates how to save data sets in two common formats. The stacked keyword creates a stacked file where all imputations are in a single file, and the separate keyword saves each imputed data set to a separate file with the asterisk replaced by a numeric index. To keep things organized, the `./imps` part of the file path points to a subfolder named `imps` located within the same folder as the script and data. The separate keyword also creates a

list of file names needed for analysis in Mplus (in this example, a filed called `implist.dat` located in the `imps` folder).

When saving imputations, the bottom of the Blimp output file displays a table listing the order of the variables in the output data sets. All variables are saved regardless of whether they appeared in the fitted models. When saving data to a stacked file (e.g., for analysis in R or other packages), the first variable in the file is an integer index that identifies which data set each row belongs to (e.g., an integer variable that ranges from 1 to 20 in this example).

```
VARIABLE ORDER IN IMPUTED DATA:

  separate = './imps/imp*.dat'

     school student condition teachexp eslpct ethnic male frlunch
     lowach stanmath efficacyp efficacy1 psolvepre psolvepst

   stacked = './imps/imps.dat'

     imp# school student condition teachexp eslpct ethnic male frlunch
     lowach stanmath efficacyp efficacy1 psolvepre psolvepst
```

The imputed data sets are subsequently analyzed in another software package, and estimates and standard errors are combined using Rubin's rules (Little & Rubin, 2020). The analysis phase does not utilize the auxiliary variables, as their information is embedded in the imputations. Scripts for analyzing the imputed data sets are found in the next subsections.

Analyzing Imputations in Mplus

In lieu of the Bayesian estimates, Blimp's `SAVE` command can be used to save multiple imputations for analysis in the frequentist framework. Returning to the previous Blimp script, the `SAVE` command and the `separate` keyword saved each imputed data set to a separate file with the asterisk replaced by a numeric index. The separate keyword also creates a list of file names needed for analysis in Mplus (in this example, a filed called `implist.dat` located in the `imps` subfolder). Example 6 shows the contents of this file.

The code block below shows Mplus script `Ex15.inp` that analyzes the imputations and pools estimates and standard errors.

Mplus Script Ex15.inp

```
1   DATA:
2   file = ./imps/implist.dat;
3   type = imputation;
4   VARIABLE:
5   names = school student condition teachexp eslpct ethnic male frlunch
6     lowach stanmath efficacy1 efficacy2 psolvepre psolvepst;
7   usevariables = psolvepst psolvepre male frlunch teachexp condition;
8   cluster = school;
9   within = psolvepre male frlunch;
10  between = teachexp condition;
11  DEFINE:
12  center psolvepre male frlunch teachexp (grandmean);
13  ANALYSIS:
14  type = twolevel;
15  MODEL:
16  %within%
17  psolvepst on psolvepre male frlunch;
18  %between%
19  psolvepst on teachexp condition;
20  OUTPUT:
21  stdyx cinterval;
```

The `DATA` command lists the text file containing the names of the imputed data sets (the `implist.dat` file located in the `./imps` subdirectory). The `type = imputation` subcommand instructs Mplus that the input data is a list of file names. The `VARIABLE` command provides information about the data. Beginning on line 5, the `names` subcommand assigns names to the variables in the input data file, and the `usevariables` subcommand selects variables for the analysis. The `cluster` command on line 8 lists the school-level identifier variable that indicates the clustering of the data records in schools. The `within` and `between` subcommands on lines 9 and 10 identify level-1 and level-2 predictors, respectively. On line 12, the `center` subcommand under the `DEFINE` command centers the four covariates at their grand means. The `ANALYSIS` command and the `type = twolevel` subcommand is required for estimating two-

level models. The `MODEL` section of the script consists of two sections: the `%within%` section specifies the regression of the outcome on level-1 predictors, and the `%between%` section specifies the regression of the random intercepts on the level-2 predictors. Finally, the `OUTPUT` command specifies two keywords on line 21 that request standardized coefficients and confidence intervals.

Mplus Output

The table of unstandardized parameter estimates is shown below. The first two columns display the pooled unstandardized estimates and standard errors, and the third and fourth columns display the corresponding *z*-statistics and *p*-values. The focal model results are shown in bold typeface. The `Rate of Missing` column (also called the fraction of missing information in the literature) quantifies the imputation noise in each estimate as proportion of its squared standard error.

```
MODEL RESULTS
                                                    Two-Tailed   Rate of
                  Estimate       S.E.  Est./S.E.    P-Value    Missing

Within Level

PSOLVEPST  ON
    PSOLVEPRE        0.464      0.036     12.766      0.000      0.382
    MALE             0.114      0.367      0.310      0.757      0.317
    FRLUNCH         -0.710      0.463     -1.534      0.125      0.423

 Residual Variances
    PSOLVEPST       20.624      1.329     15.518      0.000      0.061

Between Level

 PSOLVEPST  ON
    TEACHEXP         0.133      0.075      1.769      0.077      0.122
    CONDITION        1.850      0.822      2.251      0.024      0.048
```

```
Intercepts
    PSOLVEPST         52.879      0.533     99.185      0.000      0.051

 Residual Variances
    PSOLVEPST          3.991      1.166      3.422      0.001      0.072
```

Mplus separates the level-1 and level-2 effects on the output (labeled `Within Level` and `Between Level`, respectively). The primary focus is the β_5 coefficient, which indicates that intervention schools scored 1.85 points higher than control schools, on average, controlling for student- and school-level covariates. The corresponding test statistic indicates that the group mean difference is statistically different from zero ($z = 2.25$, $p = .02$). Note that these estimates are numerically identical to those from Bayesian estimation.

Analyzing Imputations in R

Returning to the previous Blimp script, the `SAVE` command and the `stacked` keyword saved the imputed data sets to a single stacked file with an index variable in the first column identifying the individual files. The stacked file is appropriate for analyzing data in R, SAS, SPSS, and Stata, among others.

The R input file for the analysis is `Ex12.R`. The code block below shows the commands that import the data.

R Script Ex15.R

```
1  library(fdir)
2  library(lme4)
3  library(mitml)
4  set()
5  imps <- read.table("./imps/imps.dat")
6  names(imps) <- c("imputation","school","student","condition",
7    "teachexp","eslpct","ethnic","male","frlunch","lowach",
8    "stanmath","efficacypre","efficacypst","psolvepre","psolvepst")
```

```
 9  imps$psolvepre.cgm <- imps$psolvepre - mean(imps$psolvepre)
10  imps$male.cgm <- imps$male - mean(imps$male)
11  imps$frlunch.cgm <- imps$frlunch - mean(imps$frlunch)
12  imps$teachexp.cgm <- imps$teachexp - mean(imps$teachexp)
```

The example requires the `fdir`, `lme4`, and `mitml` packages, which are loaded on lines 1 through 3. On line 4, the `set()` function of the `fdir` package identifies the file path of the folder containing the R script and sets this location as the working directory. On line 5, the `read.table` command imports the stacked data. It is only necessary to specify the name of the input data file. No file path is required when the data reside in the same folder as the R script as is the case here. Variable names are listed beginning on line 6. Importantly, the first variable named `IMPUTATION` is the index that identifies the individual files. Finally, lines 9 through 12 create new centered versions of the covariates.

The next block of code relies on the `mitml` package to fit the model to each data set and pool the results using Rubin's rules.

R Script Ex15.R, continued

```
13  implist <- as.mitml.list(split(imps, imps$imputation))
14  fit <- with(implist, lmer(psolvepst ~ psolvepre.cgm + male.cgm +
15    frlunch.cgm + teachexp.cgm + condition + (1 | school), REML = T))
16  estimates <- testEstimates(fit, extra.pars = T)
17  estimates
18  confint(estimates)
```

The `implist` command on line 13 unstacks the data and creates a list that contains the individual files. Line 14 fits the focal regression model using the `lmer` function, and line 16 uses the `testEstimates` function in `mitml` to implement Rubin's pooling rules and save the results in an object called `estimates`. Finally, lines 17 and 18 print the estimates and confidence intervals.

R Output

The table of unstandardized pooled parameter estimates is shown below. The first two columns display the pooled unstandardized estimates and standard errors, and the third through fifth columns display the corresponding test statistics. The focal model results are shown in bold typeface. The `RIV` column (relative increase in variance) is a fraction comparing imputation noise to complete-data sampling variation, and the `FMI` column (fraction of missing information in the literature) quantifies the imputation noise in each estimate as proportion of its squared standard error.

```
> estimates

Call:

testEstimates(model = fit, extra.pars = T)

Final parameter estimates and inferences obtained from 20 imputed data sets.

               Estimate Std.Error   t.value        df   P(>|t|)       RIV       FMI
(Intercept)      52.865     0.655    80.699 14410.820     0.000     0.038     0.036
psolvepre.cgm     0.463     0.038    12.057   168.177     0.000     0.506     0.344
male.cgm          0.113     0.364     0.310   189.908     0.757     0.463     0.323
frlunch.cgm      -0.701     0.495    -1.417   144.398     0.159     0.569     0.371
teachexp.cgm      0.134     0.105     1.277  4885.049     0.202     0.067     0.063
condition         1.850     0.879     2.105 10628.599     0.035     0.044     0.042

                           Estimate
Intercept~~Intercept|school    4.530
Residual~~Residual            20.690
ICC|school                     0.180

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

```
> confint(estimates)
                    2.5 %     97.5 %
(Intercept)   51.58070589 54.1488180
psolvepre.cgm  0.38737012  0.5390615
male.cgm      -0.60546360  0.8314398
frlunch.cgm   -1.68006958  0.2771710
teachexp.cgm  -0.07188544  0.3407508
condition      0.12740014  3.5730222
```

The random intercept and within-cluster residual variances are denoted `Intercept~~Intercept|school` and `Residual~~Residual`, respectively. Moving to the coefficient section, the primary focus is the β_5 coefficient, which indicates that intervention schools scored 1.85 points higher than control schools, on average, controlling for student- and school-level covariates. The corresponding test statistic indicates that the group mean difference is statistically different from zero (t = 2.11, p = .04). Note that these estimates are numerically identical to those from Bayesian estimation.

EXAMPLE 16: TWO-LEVEL REGRESSION WITH A CROSS-LEVEL INTERACTION EFFECT

This example illustrates a two-level multiple regression with random intercepts. The analysis uses the `problemsolving3level.dat` data set taken from a cluster-randomized educational intervention where 29 schools were assigned to an intervention and comparison condition. In addition to the intervention assignment indicator, school-level variables include the average years of teacher experience and the percentage of learners for whom English is a second language. The 928 student-level records include pretest and posttest math problem-solving and self-efficacy scores, standardized math scores taken from a statewide assessment, and several sociodemographic variables. The analysis variables are as follows.

Name	Definition	Missing %	Scale
	Identifier Variables		
STUDENT	Student identifier	0	Integer index
	Focal Variables		
PROBSOLVE	Math problem-solving posttest	11.5	Numeric
MONTH7	Math problem-solving pretest	0	Numeric
MALE	Gender dummy code	0	0 = Female, 1 = Male
FRLUNCH	Lunch assistance code	4.7	0 = None, 1 = Free/reduced lunch
TEACHEXP	Teacher years of experience	10.8	Numeric
CONDITION	Experimental condition	0	0 = Control, 1 = Experimental

Analysis Model

The analysis is a linear growth model that features a repeatedly-measured problem-solving test regressed on time scores (months until the end of the school year, a level-1 predictor), experimental condition (level-2), the cross-level interaction of the two variables, and three grand mean centered covariates: gender and lunch assistance dummy codes (level-1), and years of teacher experience (level-2). To convey each variable's level, the i and j subscripts denote repeated measurements and students, respectively.

$$PROBSOLVE_{ij} = (\beta_0 + b_{0j}) + (\beta_1 + b_{1j})(MONTH7_{ij}) + \beta_2(MALE_j) + \beta_3(FRLUNCH_j) + \beta_4(TEACHEXP_j) + \beta_5(CONDITION_j) + \beta_6(MONTH7_{ij})(CONDITION_j) + \varepsilon \quad (31)$$

Unlike a complete-data regression analysis, all incomplete variables require distributional assumptions, including the predictors. Blimp uses a factored regression specification that assigns separate distributions to the predictors and outcome. By default, Blimp invokes a multivariate normal distribution for numeric predictors and the latent response scores for discrete predictors.

Blimp Script

The code block below shows Blimp script `Ex16.1.inp`. The first six lines can be viewed as a set of commands that specify information about the data and variables. The `DATA` command specifies the name of the input text file. No file path is required when the data file is located in the same directory as the script, as it is here. Starting on line 2, the `VARIABLES` command names the data columns. The `CLUSTERID` command on line 4 lists the student-level identifier variable that indicates the clustering of the repeated measurements within students. Including the `CLUSTERID` command automatically introduces random intercepts. The `ORDINAL` command on line 5 identifies binary and ordinal variables. Binary variables can be defined as ordinal or nominal, as the statistical models are identical. The `MISSING` command on line 6 defines a global missing value code as 999.

Blimp Script Ex16.1.imp

```
1   DATA: problemsolving3level.dat;
2   VARIABLES: school student wave condition teachexp eslpct ethnic
3     male frlunch lowach stanmath month0 month7 probsolve efficacy;
4   CLUSTERID: student;
5   ORDINAL: male frlunch condition;
6   MISSING: 999;
7   FIXED: month7 male condition;
8   CENTER: grandmean = male frlunch teachexp;
9   MODEL:
10  probsolve ~ month7 male frlunch teachexp condition
11    month7*condition  | month7;
12  SIMPLE:
13  month7 | condition;
```

```
14  SEED: 90291;
15  BURN: 5000;
16  ITERATIONS: 20000;
```

The `FIXED`, `CENTER`, `MODEL`, and `SIMPLE` blocks can be viewed as a set. The `FIXED` command identifies a complete predictor, which does not require a distribution or regression model. The `CENTER` command deviates the three covariates at their iteratively-estimated grand means. Beginning on line 9, the `MODEL` command lists the regression model, with outcome variable to the left of the tilde and predictors to the right. The product term is specified by joining the interacting variables with an asterisk (i.e., `MONTH7*CONDITION`), and listing `MONTH7` to the right of the vertical pipe specifies this variable as a random slope predictor. The `SIMPLE` command requests the conditional effects (i.e., simple slopes) of `MONTH7` at each level of `CONDITION`. By default, Blimp computes the simple slope at each level of a binary moderator listed on the `ORDINAL` line. Blimp automatically configures the explanatory variable models under the assumption that the numeric variables and latent response scores (discrete predictors) are normally distributed. Custom significance tests can be specified using the `TEST` command, as shown in previous examples.

Finally, lines 14 through 16 can be viewed as a block of commands that specify features of the MCMC algorithm: the `SEED` command gives an integer string that initializes the random number generator, the `BURN` command specifies the number of iterations for the warm-up or burn-in period, and the `ITERATIONS` command gives the number of MCMC iterations on which the analysis summaries are based (essentially, the number of MCMC cycles following the warm-up period).

Blimp Output

Prior to inspecting the parameter estimates, it is important to investigate the potential scale reduction (PSR) factor diagnostics (Gelman & Rubin, 1992) to determine whether MCMC has converged. Blimp divides the burn-in period into 20 equal segments, and it computes the PSR diagnostic for every parameter. The table located near the top of the output reports the highest (worst) PSR value across all parameters in every model. A common recommendation is that these values should be less than 1.05 or perhaps 1.10 (Asparouhov & Muthén, 2010a; Gelman et al., 2014). If the PSR in the bottom row of the table (the final check of the burn-in period) is above these cutoffs, then rerun the analysis with a longer burn-in period.

```
BURN-IN POTENTIAL SCALE REDUCTION (PSR) OUTPUT:

  NOTE: Split chain PSR is being used. This splits each chain's
        iterations to create twice as many chains.

  Comparing iterations across 2 chains     Highest PSR   Parameter #
                          126 to 250             2.182            20
                          251 to 500             1.657             3
                          376 to 750             1.280            13
                          501 to 1000            1.145             7
                                  ...              ...            ..
                         2126 to 4250            1.017             3
                         2251 to 4500            1.034            20
                         2376 to 4750            1.053             3
                         2501 to 5000            1.033             3
```

The next section of the output displays information about the variables in the analysis and the models used for estimation. This output table mimics the one from Example 6, but it additionally reports the number of observations at each level. Earlier examples also show how to implement the Bayesian Wald significance test.

```
DATA INFORMATION:


  Level-2 identifier:       student
  Sample Size:              6874
  Level-2 Clusters:         982
  Missing Data Rates:

               probsolve = 11.45
                teachexp = 10.79
                 frlunch = 04.68
```

MCMC estimation produces a distribution for each parameter in the table. The median and standard deviation columns describe the center and spread of the posterior distributions; although they make no reference to drawing repeated samples, they are analogous—and numerically equivalent in most cases—to frequentist point estimates and standard errors. The

95% credible intervals in the rightmost columns give a range that captures 95% of the parameter's distribution. These are akin to confidence intervals, but the intervals describe parameter distributions rather than characteristics of repeated samples. The `N_Eff` values in rightmost column of the table give the effective number of MCMC samples for each parameter. These quantities essentially represent the number of independent estimates on which the parameter summaries are based after removing autocorrelations from the MCMC process. Gelman et al. (2014, p. 287) recommend values greater than 100. All values in the example table exceed this recommended minimum. In cases where the `N_Eff` values are insufficient, increasing the value on the `ITERATIONS` command will remedy the issue. Unlike previous examples, this analysis specified 20,000 iterations because the effective sample size for the random slope variance was less than 100 when using 10,000 iterations. The table summarizing the focal regression model is shown below. The table includes unstandardized coefficients, standardized slopes, and variance explained effect size estimates (Rights & Sterba, 2019).

```
OUTCOME MODEL ESTIMATES:

  Summaries based on 20000 iterations using 2 chains.


Outcome Variable:  probsolve

Grand Mean Centered: frlunch male teachexp


Parameters                        Median     StdDev       2.5%      97.5%        PSR      N_Eff
                                 ------------------------------------------------------------------
Variances:
  L2 : Var(Intercept)             17.147      1.117     15.071     19.448      1.002   1978.298
  L2 : Cov(month7,Intercept)       0.807      0.149      0.534      1.117      1.007    491.594
  L2 : Var(month7)                 0.115      0.030      0.065      0.180      1.012    216.488
  Residual Var.                   12.586      0.273     12.061     13.133      1.001   1716.711

Coefficients:
  Intercept                       52.952      0.260     52.448     53.465      1.001   1617.612
  month7                           0.433      0.042      0.352      0.517      1.000   8418.593
  male                             0.413      0.248     -0.078      0.903      1.000   1475.608
  frlunch                         -0.918      0.328     -1.542     -0.265      1.002    998.341
  teachexp                         0.021      0.033     -0.044      0.086      1.002   1043.099
  condition                        1.548      0.335      0.878      2.196      1.003   1487.024
  month7*condition                 0.348      0.053      0.245      0.452      1.000   8775.829


Standardized Coefficients:
  month7                           0.162      0.016      0.132      0.194      1.000   8412.474
  male                             0.038      0.023     -0.007      0.083      1.000   1473.307
  frlunch                         -0.068      0.024     -0.114     -0.020      1.002   1000.682
```

```
   teachexp                          0.016      0.026     -0.034      0.066      1.002   1042.047
   condition                         0.141      0.030      0.080      0.199      1.003   1509.932
   month7*condition                  0.140      0.021      0.098      0.181      1.000   8890.546

Proportion Variance Explained
   by Coefficients                   0.075      0.007      0.063      0.089      1.002   2137.040
   by Level-2 Random Intercepts      0.468      0.014      0.440      0.496      1.000   8311.275
   by Level-2 Random Slopes          0.016      0.004      0.009      0.025      1.012    216.410
   by Level-1 Residual Variation     0.441      0.013      0.415      0.467      1.001   2681.346

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

The results are interpreted in the same way as a complete-data multilevel analysis. The first section of the output table displays the variance estimates. The random intercept and slope variances are denoted `L2:Var(Intercept)` and `L2:Var(month7)`, respectively, and their covariance is labeled `L2 : Cov(month7,Intercept)`. The within-cluster residual variance is denoted `Residual Var.` Turning to the coefficients section, lower-order terms in a moderated regression are conditional effects that depend on scaling or centering. Specifically, the lower-order slope of `MONTH7` ($\beta_1 = 0.43$) is the monthly change rate for students in the comparison condition (`CONDITION` = 0), and the intervention slope ($\beta_5 = 1.55$) similarly reflects the mean difference when `MONTH7` = 0 (at the final assessment). The interaction effect captures the growth rate difference for students in experimental schools. The positive coefficient ($\beta_6 = 0.35$) indicates that the growth rate for the experimental condition is greater (more positive) than that of the comparison condition. The 95% credible interval limits suggest this effect is statistically different from zero ($p < .05$) because the null value is outside the interval. The bottom portion of the table displays Rights and Sterba (2019) *R*-squared effect size values. The fixed effects explain 7.5% of the total variation, the random intercepts account for 46.8% of the variability, and the random slopes account for 1.6% of the variation.

The `SIMPLE` command prints a table of conditional effects (simple slopes) of `MONTH7` within each intervention condition. Consistent with the positive interaction coefficient, the simple slope for the experimental schools is higher (more positive) that the growth rate for controls. Both conditional effects are statistically significant at $p < .05$ because the null value does not fall within the 95% credible intervals. The output table is shown below.

```
Conditional Effects                 Median     StdDev       2.5%      97.5%        PSR      N_Eff
                                ------------------------------------------------------------------
  month7 | condition @ 0
    Intercept                       52.952      0.260     52.448     53.465      1.001   1617.612
    Slope                            0.433      0.042      0.352      0.517      1.000   8418.593

  month7 | condition @ 1
    Intercept                       54.503      0.204     54.103     54.904      1.006   1569.880
    Slope                            0.782      0.032      0.719      0.845      1.001   8792.733

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

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

The Blimp output also includes tables of regression model parameters for the auxiliary variables as well as the auto-generated models for incomplete predictors. These additionally results are not of substantive interest and would not be reported. The auxiliary variable models appear in `OUTCOME MODEL ESTIMATES` section with the focal results, and the auto-generated predictor models are displayed under the heading `PREDICTOR MODEL ESTIMATES`.

Saving Multiple Imputations

MCMC estimation imputes missing values at every iteration, such that the resulting Bayesian estimates average over thousands of plausible replacement scores (10,000 sets in this example). A subset of the imputations can be saved for reanalysis in the frequentist framework, if desired. The Blimp input file `Ex16.2.imp` is identical `Ex16.1.imp`, but it adds the following lines at the bottom of the script.

```
NIMPS: 20;
CHAINS: 20;
SAVE:
stacked = ./imps/imps.dat;
separate = ./imps/imp*.dat;
```

The `NIMPS`, `CHAINS` and `SAVE` commands can be viewed as a set. Setting `NIMPS` equal to `CHAINS` saves a single filled-in data set from the final iteration of a unique `MCMC` process, thus avoiding autocorrelation among the imputations. The `SAVE` command provides a name for the imputed data sets. The script illustrates how to save data sets in two common formats. The

stacked keyword creates a stacked file where all imputations are in a single file, and the separate keyword saves each imputed data set to a separate file with the asterisk replaced by a numeric index. To keep things organized, the `./imps` part of the file path points to a subfolder named `imps` located within the same folder as the script and data. The separate keyword also creates a list of file names needed for analysis in Mplus (in this example, a filed called `implist.dat` located in the `imps` folder).

When saving imputations, the bottom of the Blimp output file displays a table listing the order of the variables in the output data sets. All variables are saved regardless of whether they appeared in the fitted models. When saving data to a stacked file (e.g., for analysis in R or other packages), the first variable in the file is an integer index that identifies which data set each row belongs to (e.g., an integer variable that ranges from 1 to 20 in this example).

```
VARIABLE ORDER IN IMPUTED DATA:

  separate = './imps/imp*.dat'

     school student wave condition teachexp eslpct ethnic male
     frlunch lowach stanmath month0 month7 probsolve efficacy

   stacked = './imps/imps.dat'

     imp# school student wave condition teachexp eslpct ethnic male
     frlunch lowach stanmath month0 month7 probsolve efficacy
```

The imputed data sets are subsequently analyzed in another software package, and estimates and standard errors are combined using Rubin's rules (Little & Rubin, 2020). The analysis phase does not utilize the auxiliary variables, as their information is embedded in the imputations. Scripts for analyzing the imputed data sets are found in the next subsections.

Analyzing Imputations in Mplus

In lieu of the Bayesian estimates, Blimp's `SAVE` command can be used to save multiple imputations for analysis in the frequentist framework. Returning to the previous Blimp script, the `SAVE` command and the `separate` keyword saved each imputed data set to a separate file with the asterisk replaced by a numeric index. The separate keyword also creates a list of file

names needed for analysis in Mplus (in this example, a filed called `implist.dat` located in the `imps` subfolder). Example 6 shows the contents of this file.

The Mplus input file for analyzing the imputations is `Ex13.inp`. The code block below shows the analysis and pooling script. The `DATA` command lists the text file containing the names of the imputed data sets (the `implist.dat` file located in the `./imps` subdirectory). The `type = imputation` subcommand instructs Mplus that the input data is a list of file names. The `VARIABLE` command provides information about the data. Beginning on line 5, the `names` subcommand assigns names to the variables in the input data file, and the `usevariables` subcommand selects variables for the analysis. The `cluster` command on line 8 lists the school-level identifier variable that indicates the clustering of the data records in schools. The `within` and `between` subcommands on lines 9 and 10 identify level-1 and level-2 predictors, respectively.

Mplus Script Ex16.inp

```
1   DATA:
2   file = ./imps/implist.dat;
3   type = imputation;
4   VARIABLE:
5   names = school student wave condition teachexp eslpct ethnic
6     male frlunch lowach stanmath month0 month7 probsolve efficacy;
7   usevariables = probsolve month7 male frlunch teachexp condition;
8   cluster = student;
9   within = month7 male frlunch;
10  between = male frlunch teachexp condition;
11  DEFINE:
12  center male frlunch teachexp (grandmean);
13  ANALYSIS:
14  type = twolevel random;
```

```
15  MODEL:
16  %within%
17  ranslope | probsolve on month7;
18  %between%
19  [ranslope] (beta1);
20  probsolve on male frlunch teachexp condition;
21  ranslope on condition (beta6);
22  ranslope with probsolve;
23  MODEL CONSTRAINT:
24  new(slp_cond0 slp_cond1);
25  slp_cond0 = beta1;
26  slp_cond1 = beta1 + beta6;
27  OUTPUT:
28  cinterval;
```

On line 12, the `center` subcommand under the `DEFINE` command centers the three covariates at their grand means. The `ANALYSIS` command and `type = twolevel random` subcommand is required for estimating two-level models with random slopes. The `MODEL` section of the script consists of two sections: the `%within%` section specifies the regression of the outcome on level-1 predictors, and the `%between%` section specifies the regression of the random intercepts on the level-2 predictors. In the `%within%` section, listing `ranslope` (an arbitrary name) to the left of the vertical pipe creates a level-2 latent variable capturing individual growth rates. Regressing this latent variable on `CONDITION` in the `%between%` model gives the cross-level interaction. Beginning on line 23, the `MODEL CONSTRAINT` command is used to compute conditional effects or simple slopes. First, line 24 assigns names to two new parameters (the group-specific growth rates). Lines 25 through 26 use parameter labels from the MODEL section to compute the conditional effect of `MONTH7` in each experimental group.

Mplus Output

The table of unstandardized parameter estimates is shown below. The first two columns display the pooled unstandardized estimates and standard errors, and the third and fourth columns display the corresponding *z*-statistics and *p*-values. The focal model results are shown in bold typeface. The `Rate of Missing` column (also called the fraction of missing information in the literature) quantifies the imputation noise in each estimate as proportion of its squared standard error.

```
MODEL RESULTS

                                                    Two-Tailed   Rate of
                  Estimate       S.E.  Est./S.E.    P-Value    Missing

Within Level

 Residual Variances
    PROBSOLVE        12.562      0.355     35.401      0.000      0.103

Between Level

 RANSLOPE   ON
    CONDITION         0.343      0.053      6.477      0.000      0.168

 PROBSOLVE  ON
    MALE              0.412      0.250      1.649      0.099      0.018
    FRLUNCH          -0.923      0.306     -3.011      0.003      0.071
    TEACHEXP          0.027      0.034      0.814      0.416      0.166
    CONDITION         1.527      0.334      4.565      0.000      0.102

 RANSLOPE WITH
    PROBSOLVE         0.802      0.155      5.161      0.000      0.216

 Intercepts
    PROBSOLVE        52.968      0.265    199.640      0.000      0.146
    RANSLOPE          0.438      0.041     10.674      0.000      0.215

 Residual Variances
    PROBSOLVE        16.918      1.049     16.126      0.000      0.076
    RANSLOPE          0.119      0.034      3.497      0.000      0.316

New/Additional Parameters
    SLP_COND          0.438      0.041     10.674      0.000      0.215
    SLP_COND          0.780      0.034     22.766      0.000      0.141
```

Mplus separates the level-1 and level-2 effects on the output (labeled `Within Level` and `Between Level`, respectively). Considering the coefficients, lower-order terms in a moderated

regression are conditional effects that depend on scaling or centering. Specifically, the lower-order slope of `MONTH7` ($\hat{\beta}_1 = 0.44$) is the monthly change rate for students in the comparison condition (`CONDITION` = 0), and the intervention slope ($\hat{\beta}_5 = 1.53$) similarly reflects the mean difference when `MONTH7` = 0 (at the final assessment). The interaction effect captures the growth rate difference for students in experimental schools. The positive coefficient ($\hat{\beta}_6 = 0.34$) indicates that the growth rate for the experimental condition is greater (more positive) than that of the comparison condition. The corresponding test statistic indicates that the interaction effect is statistically different from zero ($z = 6.48$, $p < .001$). Finally, the printed output also includes the table of conditional effects, which were computed using the `MODEL CONSTRAINT` command. Consistent with the positive interaction coefficient, the simple slope for the experimental schools is higher (more positive) that the growth rate for controls. Note that these estimates are numerically identical to those from Bayesian estimation.

Analyzing Imputations in R

Returning to the previous Blimp script, the `SAVE` command and the `stacked` keyword saved the imputed data sets to a single stacked file with an index variable in the first column identifying the individual files. The stacked file is appropriate for analyzing data in R, SAS, SPSS, and Stata, among others.

The R input file for the analysis is `Ex16.R`. The code block below shows the commands that import the data. The example requires the `fdir`, `lme4`, and `mitml` packages, which are loaded on lines 1 through 3. On line 4, the `set()` function of the `fdir` package identifies the file path of the folder containing the R script and sets this location as the working directory. On line 5, the `read.table` command imports the stacked data. It is only necessary to specify the name of the input data, as no file path is required when the file resides in the same folder as the R script. Variable names are listed beginning on line 6. Importantly, the first variable named `IMPUTATION` is the index that identifies the individual files. Finally, lines 9 through 11 create new centered versions of the covariates, and line 12 computes the cross-level product term.

R Script Ex16.R

```
1   library(fdir)
2   library(lme4)
3   library(mitml)
4   set()
5   imps <- read.table("./imps/imps.dat")
6   names(imps) <- c("imputation","school","student","wave","condition",
7     "teachexp","eslpct","race","male","frlunch","lowach",
8     "stanmath","month0", "month7", "probsolve", "efficacy")
9   imps$male.cgm <- imps$male - mean(imps$male)
10  imps$frlunch.cgm <- imps$frlunch - mean(imps$frlunch)
11  imps$teachexp.cgm <- imps$teachexp - mean(imps$teachexp)
12  imps$product <- imps$month7 * imps$condition
```

The next block of code relies on the `mitml` package to fit the model to each data set and pool the results using Rubin's rules.

R Script Ex16.R, continued

```
13  implist <- as.mitml.list(split(imps, imps$imputation))
14  fit <- with(implist, lmer(probsolve ~ month7  + male.cgm +
15    frlunch.cgm + teachexp.cgm + condition + product +
16    (1 + month7 | student), REML = T))
17  estimates <- testEstimates(fit, extra.pars = T)
18  estimates
19  confint(estimates)
```

The `implist` command on line 13 unstacks the data and creates a list that contains the individual files. Line 14 fits the focal regression model using the `lmer` function, and line 17 uses the `testEstimates` function in `mitml` to implement Rubin's pooling rules and save the results in an object called `estimates`. Finally, lines 18 and 19 print the estimates and confidence intervals.

The final code block below computes conditional effects or simple slopes. Lines 20 and 22 define text strings that define the computation of the conditional effect of `MONTH7` in each of the two experimental conditions, and lines 21 and 23 use the `testConstraints` function in `mitml` to compute the pooled coefficients and test statistics.

R Script Ex16.R, continued

```
20  slp_cond0 <- "month7 + product*0"
21  testConstraints(fit, constraints = slp_cond0)
22  slp_cond1 <- "month7 + product*1"
23  testConstraints(fit, constraints = slp_cond1)
```

R Output

The table of unstandardized pooled parameter estimates is shown below. The first two columns display the pooled unstandardized estimates and standard errors, and the third through fifth columns display the corresponding test statistics. The focal model results are shown in bold typeface. The `RIV` column (relative increase in variance) is a fraction comparing imputation noise to complete-data sampling variation, and the `FMI` column (fraction of missing information in the literature) quantifies the imputation noise in each estimate as proportion of its squared standard error.

Focusing on the coefficient section, lower-order terms in a moderated regression are conditional effects that depend on scaling or centering. Specifically, the lower-order slope of `MONTH7` ($\hat{\beta}_1 = 0.44$) is the monthly change rate for students in the comparison condition (`CONDITION` = 0), and the intervention slope ($\hat{\beta}_5 = 1.53$) similarly reflects the mean difference when `MONTH7` equals zero (at the final assessment). The interaction effect captures the growth rate difference for students in experimental schools. The positive coefficient ($\hat{\beta}_6 = 0.34$) indicates that the growth rate for the experimental condition is greater (more positive) than that of the comparison condition. The corresponding test statistic indicates that the interaction effect is statistically different from zero ($t = 6.36$, $p < .001$). Note that these estimates are numerically identical to those from Bayesian estimation.

```
> estimates

Call:

testEstimates(model = fit, extra.pars = T)

Final parameter estimates and inferences obtained from 20 imputed data sets.

              Estimate Std.Error   t.value        df   P(>|t|)       RIV       FMI
(Intercept)     52.968     0.268   197.999   904.060     0.000     0.170     0.147
month7           0.438     0.043    10.164   515.783     0.000     0.238     0.195
male.cgm         0.412     0.250     1.646 59548.781     0.100     0.018     0.018
frlunch.cgm     -0.923     0.321    -2.877  4605.132     0.004     0.069     0.065
teachexp.cgm     0.027     0.033     0.820   687.203     0.412     0.199     0.169
condition        1.527     0.335     4.557  1869.028     0.000     0.112     0.102
product          0.343     0.054     6.355   746.081     0.000     0.190     0.162

                            Estimate
Intercept~~Intercept|student   17.008
month7~~month7|student          0.120
Intercept~~month7|student       0.807
Residual~~Residual             12.562
ICC|student                     0.575

Unadjusted hypothesis test as appropriate in larger samples.

> confint(estimates)
                   2.5 %      97.5 %
(Intercept)  52.44296516 53.49301546
month7        0.35296167  0.52209734
male.cgm     -0.07866869  0.90255886
frlunch.cgm  -1.55140239 -0.29405478
teachexp.cgm -0.03829512  0.09324121
condition     0.86953509  2.18370698
product       0.23676192  0.44842949
```

Finally, the printed output also includes the table of conditional effects or simple slopes. Consistent with the positive interaction coefficient, the monthly growth rate for the experimental schools is higher (more positive) that the growth rate for controls.

```
testConstraints(model = fit, constraints = slp_cond0)

Hypothesis test calculated from 20 imputed data sets. The following
constraints were specified:

                         Estimate Std. Error
   month7 + product*0:      0.438      0.043

Combination method: D1

    F.value     df1     df2   P(>F)     RIV
    103.310       1 344.871   0.000   0.238

Unadjusted hypothesis test as appropriate in larger samples.
testConstraints(model = fit, constraints = slp_cond1)

Hypothesis test calculated from 20 imputed data sets. The following
constraints were specified:

                         Estimate Std. Error
   month7 + product*1:      0.780      0.033

Combination method: D1

    F.value     df1     df2   P(>F)     RIV
    548.355       1 568.584   0.000   0.174

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

EXAMPLE 17: THREE-LEVEL REGRESSION WITH A CROSS-LEVEL INTERACTION EFFECT

This example illustrates a two-level multiple regression with random intercepts. The analysis uses the `problemsolving3level.dat` data set taken from a cluster-randomized educational intervention where 29 schools were assigned to an intervention and comparison condition. In addition to the intervention assignment indicator, school-level variables include the average years of teacher experience and the percentage of learners for whom English is a second language. The 928 student-level records include pretest and posttest math problem-solving and self-efficacy scores, standardized math scores taken from a statewide assessment, and several sociodemographic variables. The analysis variables are as follows.

Name	Definition	Missing %	Scale
	Identifier Variables		
SCHOOL	School identifier	0	Integer index
STUDENT	Student identifier	0	Integer index
	Focal Variables		
PROBSOLVE	Math problem-solving posttest	11.5	Numeric
MONTH7	Math problem-solving pretest	0	Numeric
MALE	Gender dummy code	0	0 = Female, 1 = Male
FRLUNCH	Lunch assistance code	4.7	0 = None, 1 = Free/reduced lunch
TEACHEXP	Teacher years of experience	10.8	Numeric
CONDITION	Experimental condition	0	0 = Control, 1 = Experimental

Analysis Model

The analysis is a linear growth model that features a repeatedly-measured problem-solving test regressed on time scores (months until the end of the school year, a level-1 predictor), experimental condition (level-2), the cross-level interaction of the two variables, and three grand mean centered covariates: gender and lunch assistance dummy codes (level-1), and years of teacher experience (level-2). To convey each variable's level, the i and j subscripts denote repeated measurements and students, respectively, and k is the school-level identifier.

$$PROBSOLVE_{ijk} = (\beta_0 + b_{0jk} + b_{0k}) + (\beta_1 + b_{1jk} + b_{1k})(MONTH7_{ij}) + \beta_2(MALE_j) + \beta_3(FRLUNCH_j) + \beta_4(TEACHEXP_j) + \beta_5(CONDITION_j) + \beta_6(MONTH7_{ij})(CONDITION_j) + \varepsilon \quad (32)$$

Unlike a complete-data regression analysis, all incomplete variables require distributional assumptions, including the predictors. Blimp uses a factored regression specification that assigns separate distributions to the predictors and outcome. By default, Blimp invokes a multivariate normal distribution for numeric predictors and the latent response scores for discrete predictors.

Blimp Script

The code block below shows Blimp script `Ex17.1.inp`. The first six lines can be viewed as a set of commands that specify information about the data and variables. The `DATA` command specifies the name of the input text file. No file path is required when the data file is located in the same directory as the script, as it is here. Starting on line 2, the `VARIABLES` command names the data columns. The `CLUSTERID` command on line 4 lists the student- and school-level identifier variables that indicates the clustering of the data records. The order of the identifier variables does not matter. Including the `CLUSTERID` command automatically introduces random intercepts at level-2 and level-3. The `ORDINAL` command on line 5 identifies binary and ordinal variables. Binary variables can be defined as ordinal or nominal, as the statistical models are identical. The `MISSING` command on line 6 defines a global missing value code as 999.

Blimp Script Ex17.1.imp

```
1   DATA: problemsolving3level.dat;
2   VARIABLES: school student wave condition teachexp eslpct ethnic
3     male frlunch lowach stanmath month0 month7 probsolve efficacy;
4   CLUSTERID: student school;
5   ORDINAL: male frlunch condition;
6   MISSING: 999;
7   FIXED: month7 male condition;
8   CENTER: grandmean = male frlunch teachexp;
9   MODEL:
10  probsolve ~ month7 male frlunch teachexp condition
11    month7*condition  | month7;
12  SIMPLE:
13  month7 | condition;
```

```
14  SEED: 90291;
15  BURN: 20000;
16  ITERATIONS: 50000;
```

The `FIXED`, `CENTER`, `MODEL`, and `SIMPLE` blocks can be viewed as a set. The `FIXED` command identifies a complete predictor, which does not require a distribution or regression model. The `CENTER` command deviates the three covariates at their iteratively-estimated grand means. Beginning on line 9, the `MODEL` command lists the regression model, with outcome variable to the left of the tilde and predictors to the right. The product term is specified by joining the interacting variables with an asterisk (i.e., `MONTH7*CONDITION`), and listing `MONTH7` to the right of the vertical pipe specifies this variable as a random slope predictor. The `SIMPLE` command requests the conditional effects (i.e., simple slopes) of `MONTH7` at each level of `CONDITION`. By default, Blimp computes the simple slope at each level of a binary moderator listed on the `ORDINAL` line. Blimp automatically configures the explanatory variable models under the assumption that the numeric variables and latent response scores (discrete predictors) are normally distributed. Custom significance tests can be specified using the `TEST` command, as shown in previous examples.

Finally, lines 14 through 16 can be viewed as a block of commands that specify features of the MCMC algorithm: the `SEED` command gives an integer string that initializes the random number generator, the `BURN` command specifies the number of iterations for the warm-up or burn-in period, and the `ITERATIONS` command gives the number of MCMC iterations on which the analysis summaries are based (essentially, the number of MCMC cycles following the warm-up period).

Blimp Output

Prior to inspecting the parameter estimates, it is important to investigate the potential scale reduction (PSR) factor diagnostics (Gelman & Rubin, 1992) to determine whether MCMC has converged. Blimp divides the burn-in period into 20 equal segments, and it computes the PSR diagnostic for every parameter. The table located near the top of the output reports the highest (worst) PSR value across all parameters in every model. A common recommendation is that these values should be less than 1.05 or perhaps 1.10 (Asparouhov & Muthén, 2010a; Gelman et al., 2014). If the PSR in the bottom row of the table (the final check of the burn-in period) is above these cutoffs, then rerun the analysis with a longer burn-in period. This analysis required a much longer burn-in period that previous examples.

```
BURN-IN POTENTIAL SCALE REDUCTION (PSR) OUTPUT:

  NOTE: Split chain PSR is being used. This splits each chain's
        iterations to create twice as many chains.

  Comparing iterations across 2 chains     Highest PSR   Parameter #
                          501 to 1000            1.938            18
                         1001 to 2000            1.460             3
                         1501 to 3000            1.159             8
                         2001 to 4000            1.215             3
                                  ...              ...            ..
                         8501 to 17000           1.011             8
                         9001 to 18000           1.019            23
                         9501 to 19000           1.020            23
                        10001 to 20000           1.010            27
```

The next section of the output displays information about the variables in the analysis and the models used for estimation. This output table mimics the one from Example 6, but it additionally reports the number of observations at each level. Earlier examples also show how to implement the Bayesian Wald significance test.

```
DATA INFORMATION:

  Level-2 identifier:       student
  Level-3 identifier:       school
  Sample Size:              6874
  Level-2 Clusters:         982
  Level-3 Clusters:         29
```

MCMC estimation produces a distribution for each parameter in the table. The median and standard deviation columns describe the center and spread of the posterior distributions; although they make no reference to drawing repeated samples, they are analogous—and numerically equivalent in most cases—to frequentist point estimates and standard errors. The 95% credible intervals in the rightmost columns give a range that captures 95% of the parameter's distribution. These are akin to confidence intervals, but the intervals describe parameter distributions rather than characteristics of repeated samples. The `N_Eff` values in

rightmost column of the table give the effective number of MCMC samples for each parameter. These quantities essentially represent the number of independent estimates on which the parameter summaries are based after removing autocorrelations from the MCMC process. Gelman et al. (2014, p. 287) recommend values greater than 100. All values in the example table exceed this recommended minimum. In cases where the `N_Eff` values are insufficient, increasing the value on the `ITERATIONS` command will remedy the issue. Unlike previous examples, this analysis specified 50,000 iterations to achieve acceptable values. The table summarizing the focal regression model is shown below. The table includes unstandardized coefficients, standardized slopes, and variance explained effect size estimates (Rights & Sterba, 2019).

```
OUTCOME MODEL ESTIMATES:

  Summaries based on 50000 iterations using 2 chains.

Outcome Variable:  probsolve

Grand Mean Centered: frlunch male teachexp


Parameters                        Median     StdDev       2.5%      97.5%        PSR      N_Eff
                                 ------------------------------------------------------------------
Variances:
  L2 : Var(Intercept)             11.082      0.841      9.536     12.831      1.002   1518.450
  L2 : Cov(month7,Intercept)       0.319      0.118      0.110      0.569      1.015    294.347
  L2 : Var(month7)                 0.048      0.025      0.009      0.103      1.049     93.299
  L3 : Var(Intercept)              7.682      3.015      4.162     15.828      1.003   2028.329
  L3 : Cov(month7,Intercept)       0.641      0.298      0.272      1.428      1.001   4314.971
  L3 : Var(month7)                 0.093      0.039      0.047      0.196      1.000  13688.553
  Residual Var.                   12.569      0.272     12.048     13.116      1.007    805.665

Coefficients:
  Intercept                       52.947      0.821     51.210     54.439      1.009    323.296
  month7                           0.456      0.097      0.265      0.647      1.001    952.544
  male                             0.329      0.226     -0.112      0.771      1.001   4104.664
  frlunch                         -0.275      0.304     -0.870      0.320      1.001   3069.641
  teachexp                         0.008      0.093     -0.192      0.178      1.005    321.412
  condition                        1.517      1.083     -0.647      3.669      1.006    323.310
  month7*condition                 0.295      0.129      0.035      0.546      1.001    972.799
```

```
Standardized Coefficients:
  month7                           0.167      0.036      0.096      0.237      1.002    936.322
  male                             0.030      0.020     -0.010      0.070      1.002   4097.642
  frlunch                         -0.020      0.022     -0.063      0.023      1.002   3051.944
  teachexp                         0.006      0.070     -0.141      0.136      1.005    323.493
```

```
  condition                          0.136      0.094     -0.057      0.318      1.006    324.526
  month7*condition                   0.116      0.050      0.014      0.211      1.001   1010.021

Proportion Variance Explained
  by Coefficients                    0.072      0.017      0.045      0.114      1.005    536.434
  by Level-2 Random Intercepts       0.323      0.024      0.269      0.365      1.004   1830.280
  by Level-2 Random Slopes           0.006      0.003      0.001      0.014      1.048     96.338
  by Level-3 Random Intercepts       0.157      0.047      0.092      0.275      1.003   1823.019
  by Level-3 Random Slopes           0.013      0.005      0.006      0.025      1.000  14140.962
  by Level-1 Residual Variation      0.424      0.028      0.357      0.469      1.007   1356.065

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

The results are interpreted in the same way as a complete-data multilevel analysis. The first section of the output table displays the variance estimates. The level-2 random intercept and slope variances are denoted `L2:Var(Intercept)` and `L2:Var(month7)`, respectively, and their covariance is labeled `L2 : Cov(month7,Intercept)`. Similarly, the level-3 random intercept and slope variances are denoted `L3:Var(Intercept)` and `L3:Var(month7)`, respectively, and their covariance is labeled `L3 : Cov(month7,Intercept)`. The within-cluster residual variance is denoted `Residual Var.` Turning to the coefficients section, lower-order terms in a moderated regression are conditional effects that depend on scaling or centering. Specifically, the lower-order slope of `MONTH7` (β_1 = 0.47) is the monthly change rate for students in the comparison condition (`CONDITION` = 0), and the intervention slope (β_5 = 1.52) similarly reflects the mean difference when `MONTH7` = 0 (at the final assessment). The interaction effect captures the growth rate difference for students in experimental schools. The positive coefficient (β_6 = 0.30) indicates that the growth rate for the experimental condition is greater (more positive) than that of the comparison condition. The 95% credible interval limits suggest this effect is statistically different from zero ($p < .05$) because the null value is outside the interval. The bottom portion of the table displays Rights and Sterba (2019) *R*-squared effect size values. The fixed effects explain 7.2% of the total variation, the random intercepts at level-2 and level-3 account for 32.3% and 15.7% of the variability, respectively, and the level-2 and level-3 random slopes account for 0.6% and 1.6% of the variation.

The `SIMPLE` command prints a table of conditional effects (simple slopes) of `MONTH7` within each intervention condition. Consistent with the positive interaction coefficient, the simple slope for the experimental schools is higher (more positive) that the growth rate for controls. Both conditional effects are statistically significant at $p < .05$ because the null value does not fall within the 95% credible intervals. The output table is shown below.

```
Conditional Effects                Median     StdDev       2.5%      97.5%        PSR      N_Eff
                                  ----------------------------------------------------------------
```

```
month7 | condition @ 0
  Intercept                    52.947      0.821     51.210     54.439      1.009    323.296
  Slope                         0.456      0.097      0.265      0.647      1.001    952.544

month7 | condition @ 1
  Intercept                    54.437      0.765     52.894     55.929      1.002    209.317
  Slope                         0.751      0.086      0.577      0.917      1.001    720.410

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

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

The Blimp output also includes tables of regression model parameters for the auxiliary variables as well as the auto-generated models for incomplete predictors. These additionally results are not of substantive interest and would not be reported. The auxiliary variable models appear in `OUTCOME MODEL ESTIMATES` section with the focal results, and the auto-generated predictor models are displayed under the heading `PREDICTOR MODEL ESTIMATES`.

Saving Multiple Imputations

MCMC estimation imputes missing values at every iteration, such that the resulting Bayesian estimates average over thousands of plausible replacement scores (10,000 sets in this example). A subset of the imputations can be saved for reanalysis in the frequentist framework, if desired. The Blimp input file `Ex17.2.imp` is identical `Ex17.1.imp`, but it adds the following lines at the bottom of the script.

```
NIMPS: 20;
CHAINS: 20;
SAVE:
stacked = ./imps/imps.dat;
separate = ./imps/imp*.dat;
```

The `NIMPS`, `CHAINS` and `SAVE` commands can be viewed as a set. Setting `NIMPS` equal to `CHAINS` saves a single filled-in data set from the final iteration of a unique `MCMC` process, thus avoiding autocorrelation among the imputations. The `SAVE` command provides a name for the imputed data sets. The script illustrates how to save data sets in two common formats. The stacked keyword creates a stacked file where all imputations are in a single file, and the separate keyword saves each imputed data set to a separate file with the asterisk replaced by a numeric

index. To keep things organized, the `./imps` part of the file path points to a subfolder named `imps` located within the same folder as the script and data. The separate keyword also creates a list of file names needed for analysis in Mplus (in this example, a filed called `implist.dat` located in the `imps` folder).

When saving imputations, the bottom of the Blimp output file displays a table listing the order of the variables in the output data sets. All variables are saved regardless of whether they appeared in the fitted models. When saving data to a stacked file (e.g., for analysis in R or other packages), the first variable in the file is an integer index that identifies which data set each row belongs to (e.g., an integer variable that ranges from 1 to 20 in this example).

```
VARIABLE ORDER IN IMPUTED DATA:

  separate = './imps/imp*.dat'

     school student wave condition teachexp eslpct ethnic male
     frlunch lowach stanmath month0 month7 probsolve efficacy

   stacked = './imps/imps.dat'

     imp# school student wave condition teachexp eslpct ethnic male
     frlunch lowach stanmath month0 month7 probsolve efficacy
```

The imputed data sets are subsequently analyzed in another software package, and estimates and standard errors are combined using Rubin's rules (Little & Rubin, 2020). The analysis phase does not utilize the auxiliary variables, as their information is embedded in the imputations. Scripts for analyzing the imputed data sets are found in the next subsections.

Analyzing Imputations in Mplus

In lieu of the Bayesian estimates, Blimp's `SAVE` command can be used to save multiple imputations for analysis in the frequentist framework. Returning to the previous Blimp script, the `SAVE` command and the `separate` keyword saved each imputed data set to a separate file with the asterisk replaced by a numeric index. The separate keyword also creates a list of file names needed for analysis in Mplus (in this example, a filed called `implist.dat` located in the `imps` subfolder). Example 6 shows the contents of this file.

The Mplus input file for analyzing the imputations is `Ex17.inp`. The code block below shows the analysis and pooling script. The `DATA` command lists the text file containing the names of the imputed data sets (the `implist.dat` file located in the `./imps` subdirectory). The `type = imputation` subcommand instructs Mplus that the input data is a list of file names. The `VARIABLE` command provides information about the data. Beginning on line 5, the `names` subcommand assigns names to the variables in the input data file, and the `usevariables` subcommand selects variables for the analysis. The `cluster` subcommand on line 8 lists the school- and student-level identifier variables that indicate the clustering of the data records. The `within` and `between` subcommands on lines 9 and 10 identify level-1, level-2 ,and level-3 predictors.

Mplus Script Ex17.inp

```
1   DATA:
2   file = ./imps/implist.dat;
3   type = imputation;
4   VARIABLE:
5   names = school student wave condition teachexp eslpct ethnic
6     male frlunch lowach stanmath month0 month7 probsolve efficacy;
7   usevariables = probsolve month7 male frlunch teachexp condition;
8   cluster = school student;
9   within = month7;
10  between = (student) male frlunch (school) teachexp condition;
11  DEFINE:
12  center male frlunch teachexp (grandmean);
13  ANALYSIS:
14  type = threelevel random;
```

```
15  MODEL:
16  %within%
17  ranslope | probsolve on month7;
18  %between student%
19  probsolve on male frlunch;
20  probsolve with ranslope;
21  %between school%
```

```
22  [ranslope] (beta1);
23  probsolve on teachexp condition;
24  ranslope on condition (beta6);
25  ranslope with probsolve;
26  MODEL CONSTRAINT:
27  new(slp_cond0 slp_cond1);
28  slp_cond0 = beta1;
29  slp_cond1 = beta1 + beta6;
30  OUTPUT:
31  cinterval;
```

On line 12, the `center` subcommand under the `DEFINE` command centers the three covariates at their grand means. The `ANALYSIS` command and `type = threelevel random` subcommand is required for estimating three-level models with random slopes at each level. The `MODEL` section of the script consists of three sections: the `%within%` section specifies the regression of the outcome on level-1 time scores, and the `%between student%` section specifies the regression of the random intercepts on the level-2 predictors, and the `%between school%` section specifies the regression of the random intercepts on the level-3 predictors. In the `%within%` section, listing `ranslope` (an arbitrary name) to the left of the vertical pipe creates level-2 and level-3 latent variable capturing growth rates. Regressing this latent variable on `CONDITION` in the `%between school%` model gives the cross-level interaction. Beginning on line 26, the `MODEL CONSTRAINT` command is used to compute conditional effects or simple slopes. First, line 27 assigns names to two new parameters (the group-specific growth rates). Lines 28 and 29 use parameter labels from the `MODEL` section to compute the conditional effect of `MONTH7` in each experimental condition.

Mplus Output

The table of unstandardized parameter estimates is shown below. The first two columns display the pooled unstandardized estimates and standard errors, and the third and fourth columns display the corresponding *z*-statistics and *p*-values. The focal model results are shown in bold typeface. The `Rate of Missing` column (also called the fraction of missing information in the literature) quantifies the imputation noise in each estimate as proportion of its squared standard error.

```
MODEL RESULTS

                                                    Two-Tailed   Rate of
                   Estimate       S.E.  Est./S.E.    P-Value    Missing
Within Level

Residual Variances
    PROBSOLVE        12.563      0.812     15.467      0.000      0.013

Between STUDENT Level

 PROBSOLVE  ON
    MALE              0.336      0.258      1.301      0.193      0.017
    FRLUNCH          -0.304      0.308     -0.986      0.324      0.084

 PROBSOLV WITH
    RANSLOPE          0.300      0.144      2.084      0.037      0.101

 Variances
    RANSLOPE          0.042      0.028      1.464      0.143      0.143

 Residual Variances
    PROBSOLVE        10.970      0.827     13.272      0.000      0.098

Between SCHOOL Level

 RANSLOPE   ON
    CONDITION         0.301      0.113      2.663      0.008      0.058
```

```
 PROBSOLVE  ON
    TEACHEXP           0.014      0.073      0.191      0.848      0.033
    CONDITION          1.558      0.949      1.642      0.101      0.015

 RANSLOPE WITH
    PROBSOLVE          0.490      0.180      2.717      0.007      0.052

 Intercepts
    PROBSOLVE         52.855      0.735     71.869      0.000      0.020
    RANSLOPE           0.449      0.080      5.588      0.000      0.083

 Residual Variances
    PROBSOLVE          5.696      1.905      2.989      0.003      0.026
    RANSLOPE           0.070      0.021      3.336      0.001      0.063

New/Additional Parameters
    SLP_COND           0.449      0.080      5.588      0.000      0.083
    SLP_COND           0.749      0.079      9.545      0.000      0.017
```

Mplus separates level-specific effects on the output (labeled `Within Level` and `Between STUDENT Level`, and `Between SCHOOL Level`). Considering the coefficients, lower-order terms in a moderated regression are conditional effects that depend on scaling or centering. Specifically, the lower-order slope of `MONTH7` ($\hat{\beta}_1 = 0.45$) is the monthly change rate for students in the comparison condition (`CONDITION` = 0), and the intervention slope ($\hat{\beta}_5 = 1.56$) similarly reflects the mean difference when `MONTH7` = 0 (at the final assessment). The interaction effect captures the growth rate difference for students in experimental schools. The positive coefficient ($\hat{\beta}_6 = 0.30$) indicates that the growth rate for the experimental condition is greater (more positive) than that of the comparison condition. The corresponding test statistic indicates that the interaction is statistically different from zero ($z = 2.66$, $p = .01$). Finally, the printed output also includes the table of conditional effects, which were computed using the `MODEL CONSTRAINT` command. Consistent with the positive interaction coefficient, the simple slope for the experimental schools is higher (more positive) that the growth rate for controls. Note that these estimates are numerically identical to those from Bayesian estimation.

Analyzing Imputations in R

Returning to the previous Blimp script, the `SAVE` command and the `stacked` keyword saved the imputed data sets to a single stacked file with an index variable in the first column identifying the individual files. The stacked file is appropriate for analyzing data in R, SAS, SPSS, and Stata, among others.

The R input file for the analysis is `Ex14.R`. The code block below shows the commands that import the data. The example requires the `fdir`, `lme4`, and `mitml` packages, which are loaded on lines 1 through 3. On line 4, the `set()` function of the `fdir` package identifies the file path of the folder containing the R script and sets this location as the working directory. On line 5, the `read.table` command imports the stacked data. It is only necessary to specify the name of the input data, as no file path is required when the file resides in the same folder as the R script. Variable names are listed beginning on line 6. Importantly, the first variable named `IMPUTATION` is the index that identifies the individual files. Finally, lines 9 through 11 create new centered versions of the covariates, and line 12 computes the cross-level product term.

R Script Ex17.R

```
1   library(fdir)
2   library(lme4)
3   library(mitml)
4   set()
5   imps <- read.table("./imps/imps.dat")
6   names(imps) <- c("imputation","school","student","wave","condition",
7     "teachexp","eslpct","race","male","frlunch","lowach",
8     "stanmath","month0", "month7", "probsolve", "efficacy")
9   imps$male.cgm <- imps$male - mean(imps$male)
10  imps$frlunch.cgm <- imps$frlunch - mean(imps$frlunch)
11  imps$teachexp.cgm <- imps$teachexp - mean(imps$teachexp)
12  imps$product <- imps$month7 * imps$condition
```

The next block of code relies on the `mitml` package to fit the model to each data set and pool the results using Rubin's rules. The `implist` command on line 13 unstacks the data and creates a list that contains the individual files. Line 14 fits the focal regression model using the `lmer` function, and line 17 uses the `testEstimates` function in `mitml` to implement Rubin's pooling

rules and save the results in an object called `estimates`. Finally, lines 18 and 19 print the estimates and confidence intervals.

R Script Ex17.R, continued

```
13  implist <- as.mitml.list(split(imps, imps$imputation))
14  fit <- with(implist, lmer(probsolve ~ month7  + male.cgm +
15    frlunch.cgm + teachexp.cgm + condition + product +
16    (1 + month7 | school/student), REML = T))
17  estimates <- testEstimates(fit, extra.pars = T)
18  estimates
19  confint(estimates)
```

The final code block below computes conditional effects or simple slopes. Lines 20 and 22 define text strings that define the computation of the conditional effect of `MONTH7` in each of the two experimental conditions, and lines 21 and 23 use the `testConstraints` function in `mitml` to compute the pooled coefficients and test statistics.

R Script Ex17.R, continued

```
20  slp_cond0 <- "month7 + product*0"
21  testConstraints(fit, constraints = slp_cond0)
22  slp_cond1 <- "month7 + product*1"
23  testConstraints(fit, constraints = slp_cond1)
```

R Output

The table of unstandardized pooled parameter estimates is shown below. The first two columns display the pooled unstandardized estimates and standard errors, and the third through fifth columns display the corresponding test statistics. The focal model results are shown in bold typeface. The `RIV` column (relative increase in variance) is a fraction comparing imputation noise to complete-data sampling variation, and the `FMI` column (fraction of missing information in the literature) quantifies the imputation noise in each estimate as proportion of its squared standard error.

```
> estimates

Call:

mitml::testEstimates(model = fit, extra.pars = T)

Final parameter estimates and inferences obtained from 20 imputed data sets.

              Estimate Std.Error   t.value        df   P(>|t|)       RIV       FMI
(Intercept)     52.850     0.739    71.484 48272.908     0.000     0.020     0.020
month7           0.449     0.089     5.051  4118.663     0.000     0.073     0.068
male.cgm         0.336     0.225     1.491 36139.465     0.136     0.023     0.023
frlunch.cgm     -0.296     0.306    -0.968  2653.706     0.333     0.092     0.085
teachexp.cgm     0.014     0.080     0.175 23752.427     0.861     0.029     0.028
condition        1.558     0.988     1.577 97549.706     0.115     0.014     0.014
product          0.299     0.118     2.545  6592.702     0.011     0.057     0.054

                                    Estimate
Intercept~~Intercept|student:school   10.996
month7~~month7|student:school          0.042
Intercept~~month7|student:school       0.300
Intercept~~Intercept|school            6.279
month7~~month7|school                  0.077
Intercept~~month7|school               0.532
Residual~~Residual                    12.563

Unadjusted hypothesis test as appropriate in larger samples.

> confint(estimates)
                   2.5 %     97.5 %
(Intercept)  51.40094167 54.2991096
month7        0.27493237  0.6237389
male.cgm     -0.10566298  0.7767882
frlunch.cgm  -0.89628585  0.3036326
teachexp.cgm -0.14236201  0.1702211
condition    -0.37855824  3.4938980
product       0.06871323  0.5297675
```

Focusing on the coefficient section, lower-order terms in a moderated regression are conditional effects that depend on scaling or centering. Specifically, the lower-order slope of `MONTH7` ($\hat{\beta}_1 = 0.45$) is the monthly change rate for students in the comparison condition (`CONDITION` = 0), and the intervention slope ($\hat{\beta}_5 = 1.56$) similarly reflects the mean difference when `MONTH7` equals zero (at the final assessment). The interaction effect captures the growth rate difference for students in experimental schools. The positive coefficient ($\hat{\beta}_6 = 0.30$) indicates that the growth rate for the experimental condition is greater (more positive) than that of the comparison condition. The corresponding test statistic indicates that the interaction effect is statistically different from zero ($t = 2.55$, $p = .01$). Note that these estimates are numerically identical to those from Bayesian estimation.

Finally, the printed output also includes the table of conditional effects or simple slopes. Consistent with the positive interaction coefficient, the monthly growth rate for the experimental schools is higher (more positive) that the growth rate for controls.

```
testConstraints(model = fit, constraints = slp_cond0)

Hypothesis test calculated from 20 imputed data sets. The following
constraints were specified:

                         Estimate Std. Error
   month7 + product*0:      0.449      0.089

Combination method: D1

     F.value      df1      df2    P(>F)      RIV
      25.514        1 2648.828    0.000    0.073

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

```
testConstraints(model = fit, constraints = slp_cond1)

Hypothesis test calculated from 20 imputed data sets. The following
constraints were specified:

                         Estimate Std. Error
   month7 + product*1:      0.749      0.076

Combination method: D1

      F.value       df1       df2     P(>F)       RIV
       96.416         1 38733.604     0.000     0.018

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

EXAMPLE 18: FULLY CONDITIONAL SPECIFICATION IMPUTATION FOR MULTILEVEL MODELS WITH RANDOM INTERCEPTS

This example illustrates model-agnostic fully conditional specification multiple imputation for multilevel data with random intercepts. The analysis uses the `problemsolving2level.dat` data set taken from a cluster-randomized educational intervention where 29 schools were assigned to an intervention and comparison condition. In addition to the intervention assignment indicator, school-level variables include the average years of teacher experience and the percentage of learners for whom English is a second language. The 928 student-level records include pretest and posttest math problem-solving and self-efficacy scores, standardized math scores taken from a statewide assessment, and several sociodemographic variables. The analysis variables are as follows.

Name	Definition	Missing %	Scale
	Identifier Variables		
SCHOOL	School identifier	0	Integer index
	Focal Variables		
PSOLVEPRE	Math problem-solving pretest	0	Numeric
MALE	Gender dummy code	0	0 = Female, 1 = Male
FRLUNCH	Lunch assistance code	4.7	0 = None, 1 = Free/reduced lunch
TEACHEXP	Teacher years of experience	10.3	Numeric
CONDITION	Experimental condition	0	0 = Control, 1 = Experimental
	Auxiliary Variables		
LOWACH	Low achievement code	2.1	0 = Typical, 1 = Low achieving
STANMATH	Standardized math scores	7.4	Numeric

Imputation and Analysis Models

Fully conditional specification uses a sequence of regression models to fill in missing values. Specifically, each MCMC iteration fits a series of models where one incomplete variable is regressed on all other variables. The predicted values and residual variance from each model define the center and spread of the imputed values, which are drawn at random from a normal distribution. After imputing the missing scores, the filled-in variable becomes a predictor in all

other imputation models in the sequence. The imputation stage should include all variables and effects for the subsequent analyses, and it should incorporate auxiliary variables that either predict missingness or correlate with the incomplete variables (Collins et al., 2001).

A common goal of model-agnostic imputation is to generate imputations for different purposes (e.g., descriptive summaries, several analyses within the same project). To illustrate an entire multiple imputation analysis, suppose that one use of the filled-in data sets involves a random intercept regression model featuring problem-solving posttest scores regressed on the experimental condition dummy code at level-2 and four covariates, all of which are grand mean centered: problem-solving pretest scores (level-1), gender and lunch assistance dummy codes (level-1), and years of teacher experience (level-2). To convey each variable's level, the *i* and *j* subscripts denote students and schools, respectively.

$$\begin{aligned} PSOLVEPST_{ij} &= (\beta_0 + b_{0j}) + \beta_1(PSOLVEPRE_{ij}) + \beta_2(MALE_{ij}) \\ &+ \beta_3(FRLUNCH_{ij}) + \beta_4(TEACHEXP_j) + \beta_5(CONDITION_j) + \varepsilon_{ij} \end{aligned} \tag{33}$$

Example 12 used the same analysis model to illustrate Bayesian estimation and model-based multiple imputation.

Blimp Script

The code block below shows Blimp script `Ex18.inp`. The first six lines can be viewed as a set of commands that specify information about the data and variables. The `DATA` command specifies the name of the input text file. No file path is required when the data file is located in the same directory as the script, as it is here. Starting on line 2, the `VARIABLES` command names the data columns. The `CLUSTERID` command on line 4 lists the school-level identifier variable that indicates the clustering of the data records in schools. Including the `CLUSTERID` command automatically introduces random intercepts for all level-1 variables. When a level-1 variable appears as a predictor of another level-1 variable, its random intercepts are used as a level-2 covariate in the imputation model (i.e., imputation uses latent contextual effects). When a level-1 variable appears as a predictor of a level-2 variable, just the random intercepts are in the imputation model. The `ORDINAL` command on line 5 identifies binary and ordinal variables. Binary variables can be defined as ordinal or nominal, as the statistical models are identical. The `MISSING` command on line 6 defines a global missing value code as 999.

Blimp Script Ex18.imp

```
1  DATA: problemsolving2level.dat;
2  VARIABLES: school student condition teachexp eslpct ethnic male
3    frlunch lowach stanmath efficacypre efficacypst psolvepre psolvepst;
4  CLUSTERID: school;
5  ORDINAL: condition male frlunch lowach;
6  MISSING: 999;
7  FIXED: condition male psolvepre;
8  FCS: psolvepst psolvepre male frlunch teachexp condition
9     stanmath lowach;
10 SEED: 90291;
11 BURN: 3000;
12 ITERATIONS: 10000;
13 NIMPS: 20;
14 CHAINS: 20;
15 SAVE:
16 stacked = ./imps/imps.dat;
17 separate = ./imps/imp*.dat;
```

Next, the `FCS` command lists all variables—complete or incomplete at either level—that are included in the imputation phase. Using the `FIXED` command to identify complete variables reduces computational time because these variables do not require a regression model (see Example 17). Lines 10 through 12 can also be viewed as a block of commands that specify features of the MCMC algorithm: the `SEED` command gives an integer string that initializes the random number generator, the `BURN` command specifies the number of iterations for the warm-up or burn-in period, and the `ITERATIONS` command gives the number of MCMC iterations on which the imputation model summaries are based (essentially, the total number of MCMC cycles across all chains following the warm-up period).

The `NIMPS`, `CHAINS` and `SAVE` commands can be viewed as a set. Setting `NIMPS` equal to `CHAINS` saves a single filled-in data set from the final iteration of a unique `MCMC` process, thus avoiding autocorrelation among the imputations. The `SAVE` command provides a name for the imputed data sets. The script illustrates how to save data sets in two common formats. The stacked keyword creates a stacked file where all imputations are in a single file, and the separate keyword saves each imputed data set to a separate file with the asterisk replaced by a numeric index. To keep things organized, the `./imps` part of the file path points to a subfolder named `imps` located within the same folder as the script and data. The separate keyword also creates a

list of file names needed for analysis in Mplus (in this example, a filed called `implist.dat` located in the `imps` folder).

Blimp Output

Prior to inspecting the parameter estimates, it is important to investigate the potential scale reduction (PSR) factor diagnostics (Gelman & Rubin, 1992) to determine whether MCMC has converged. Blimp divides the burn-in period into 20 equal segments, and it computes the PSR diagnostic for every parameter. The table located near the top of the output reports the highest (worst) PSR value across all parameters in every model. A common recommendation is that these values should be less than 1.05 or perhaps 1.10 (Asparouhov & Muthén, 2010a; Gelman et al., 2014). If the PSR in the bottom row of the table (the final check of the burn-in period) is above these cutoffs, then rerun the analysis with a longer burn-in period.

```
BURN-IN POTENTIAL SCALE REDUCTION (PSR) OUTPUT:

  NOTE: Split chain PSR is being used. This splits each chain's
        iterations to create twice as many chains.

  Comparing iterations across 2 chains     Highest PSR   Parameter #
                            51 to 100           1.490            24
                           101 to 200           1.306            24
                           151 to 300           1.146            24
                           201 to 400           1.099            24
                                  ...             ...            ..
                           851 to 1700          1.022            24
                           901 to 1800          1.020            24
                           951 to 1900          1.020            17
                          1001 to 2000          1.020            17
```

The next output excerpt shows information about the data and the variables in the imputation models.

```
DATA INFORMATION:

  Level-2 identifier:       school
```

```
  Sample Size:              982
  Level-2 Clusters:         29
  Missing Data Rates:

                teachexp = 10.34
                 frlunch = 04.68
                   lowach = 02.14
                stanmath = 07.43
               psolvepst = 20.47

VARIABLES IN IMPUTATION MODEL:

  Fixed variables:        condition male psolvepre
  Incomplete continuous:  teachexp stanmath psolvepst
  Incomplete ordinal:     frlunch lowach

  NUMBER OF PARAMETERS
  Imputation Models:      57
```

MCMC estimation produces a distribution for each parameter in every unique imputation model. The median and standard deviation columns describe the center and spread of the posterior distributions; although they make no reference to drawing repeated samples, they are analogous—and numerically equivalent in most cases—to frequentist point estimates and standard errors. The 95% credible intervals in the rightmost columns give a range that captures 95% of the parameter's distribution. These are akin to confidence intervals, but the intervals describe parameter distributions rather than characteristics of repeated samples. The `N_Eff` values in rightmost column of the table give the effective number of MCMC samples for each parameter. These quantities essentially represent the number of independent estimates on which the parameter summaries are based after removing autocorrelations from the MCMC process. Gelman et al. (2014, p. 287) recommend values greater than 100. All values in the example table exceed this recommended minimum. In cases where the `N_Eff` values are insufficient, increasing the value on the `ITERATIONS` command will remedy the issue.

The Blimp output includes tables of regression parameters for every incomplete variable's imputation model. The imputation model parameters are not of substantive interest and would not be reported. An example table is shown below.

```
Missing variable:   stanmath
```

```
Parameters                      Median     StdDev       2.5%      97.5%        PSR      N_Eff
                          ----------------------------------------------------------------------

Grand Mean                      50.368      0.800     48.832     51.925      1.020   1086.118

Level 1:
  male                           1.475      0.574      0.371      2.607      1.006   4742.049
  frlunch                       -0.731      0.411     -1.512      0.094      1.006   2248.736
  lowach                         0.742      0.443     -0.138      1.589      1.011   1967.506
  psolvepre                      0.755      0.062      0.633      0.877      1.005   5383.972
  psolvepst                      0.523      0.063      0.398      0.643      1.004   5796.847
  Residual Var.                 54.358      2.780     49.104     60.076      1.003   5982.807

Level 2:
  condition                     -1.988      1.219     -4.409      0.450      1.005   2832.231
  teachexp                       0.009      0.145     -0.286      0.295      1.011   2501.374
  frlunch                       -2.089      1.307     -4.659      0.466      1.016   1607.934
  lowach                         2.339      1.770     -1.333      5.652      1.019   1217.662
  psolvepst                      0.535      0.305     -0.060      1.137      1.008   2187.522
  Residual Var.                  3.991      2.116      1.296      9.440      1.019   1395.242

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

When saving imputations, the bottom of the Blimp output file displays a table listing the order of the variables in the output data sets. All variables are saved regardless of whether they appeared in the fitted models. When saving data to a stacked file (e.g., for analysis in R or other packages), the first variable in the file is an integer index that identifies which data set each row belongs to (e.g., an integer variable that ranges from 1 to 20 in this example).

```
VARIABLE ORDER IN IMPUTED DATA:

  separate = './imps/imp*.dat'

     school student condition teachexp eslpct ethnic male frlunch
     lowach stanmath efficacyp efficacy1 psolvepre psolvepst

   stacked = './imps/imps.dat'

     imp# school student condition teachexp eslpct ethnic male frlunch
     lowach stanmath efficacyp efficacy1 psolvepre psolvepst
```

The imputed data sets are subsequently analyzed in another software package, and estimates and standard errors are combined using Rubin's rules (Little & Rubin, 2020). The analysis phase does not utilize the auxiliary variables, as their information is embedded in the imputations. Scripts for analyzing the imputed data sets are found in the next subsections.

Analyzing Imputations in Mplus

In lieu of the Bayesian estimates, Blimp's `SAVE` command can be used to save multiple imputations for analysis in the frequentist framework. Returning to the previous Blimp script, the `SAVE` command and the `separate` keyword saved each imputed data set to a separate file with the asterisk replaced by a numeric index. The separate keyword also creates a list of file names needed for analysis in Mplus (in this example, a filed called `implist.dat` located in the `imps` subfolder). Example 6 shows the contents of this file.

The code block below shows Mplus script `Ex18.inp` that analyzes the imputations and pools estimates and standard errors.

Mplus Script Ex18.inp

```
DATA:
file = ./imps/implist.dat;
type = imputation;
VARIABLE:
names = school student condition teachexp eslpct ethnic male frlunch
  lowach stanmath efficacy1 efficacy2 psolvepre psolvepst;
usevariables = psolvepst psolvepre male frlunch teachexp condition;
cluster = school;
within = psolvepre male frlunch;

between = teachexp condition;
DEFINE:
center psolvepre male frlunch teachexp (grandmean);
ANALYSIS:
type = twolevel;
MODEL:
%within%
psolvepst on psolvepre male frlunch;
%between%
```

```
19   psolvepst on teachexp condition;
20   OUTPUT:
21   stdyx cinterval;
```

The `DATA` command lists the text file containing the names of the imputed data sets (the `implist.dat` file located in the `./imps` subdirectory). The `type = imputation` subcommand instructs Mplus that the input data is a list of file names. The `VARIABLE` command provides information about the data. Beginning on line 5, the `names` subcommand assigns names to the variables in the input data file, and the `usevariables` subcommand selects variables for the analysis. The `cluster` command on line 8 lists the school-level identifier variable that indicates the clustering of the data records in schools. The `within` and `between` subcommands on lines 9 and 10 identify level-1 and level-2 predictors, respectively. On line 12, the `center` subcommand under the `DEFINE` command centers the four covariates at their grand means. The `ANALYSIS` command and the `type = twolevel` subcommand is required for estimating two-level models. The `MODEL` section of the script consists of two sections: the `%within%` section specifies the regression of the outcome on level-1 predictors, and the `%between%` section specifies the regression of the random intercepts on the level-2 predictors. Finally, the `OUTPUT` command specifies two keywords on line 21 that request standardized coefficients and confidence intervals.

Mplus Output

The table of unstandardized parameter estimates is shown below. The first two columns display the pooled unstandardized estimates and standard errors, and the third and fourth columns display the corresponding z-statistics and p-values. The focal model results are shown in bold typeface. The `Rate of Missing` column (also called the fraction of missing information in the literature) quantifies the imputation noise in each estimate as proportion of its squared standard error.

```
MODEL RESULTS
                                                    Two-Tailed   Rate of
                  Estimate       S.E.  Est./S.E.    P-Value    Missing

Within Level

 PSOLVEPST  ON
```

```
    PSOLVEPRE          0.467      0.031     15.234      0.000      0.163
    MALE               0.155      0.348      0.445      0.656      0.187
    FRLUNCH           -0.733      0.431     -1.700      0.089      0.351

 Residual Variances
    PSOLVEPST         20.497      1.351     15.168      0.000      0.150

Between Level

 PSOLVEPST  ON
    TEACHEXP           0.133      0.074      1.784      0.074      0.158
    CONDITION          1.840      0.813      2.263      0.024      0.040

 Intercepts
    PSOLVEPST         52.874      0.532     99.423      0.000      0.049

 Residual Variances
    PSOLVEPST          3.934      1.160      3.392      0.001      0.048
```

Mplus separates the level-1 and level-2 effects on the output (labeled `Within Level` and `Between Level`, respectively). The primary focus is the β_5 coefficient, which indicates that intervention schools scored 1.84 points higher than control schools, on average, controlling for student- and school-level covariates. The corresponding test statistic indicates that the group mean difference is statistically different from zero ($z = 2.26$, $p = .02$). Note that these estimates are numerically identical to those from Bayesian estimation.

Analyzing Imputations in R

Returning to the previous Blimp script, the `SAVE` command and the `stacked` keyword saved the imputed data sets to a single stacked file with an index variable in the first column identifying the individual files. The stacked file is appropriate for analyzing data in R, SAS, SPSS, and Stata, among others.

The R input file for the analysis is `Ex18.R`. The code block below shows the commands that import the data.

R Script Ex18.R

```
1   library(fdir)
2   library(lme4)
3   library(mitml)
4   set()
5   imps <- read.table("./imps/imps.dat")
6   names(imps) <- c("imputation","school","student","condition",
7     "teachexp","eslpct","ethnic","male","frlunch","lowach",
8     "stanmath","efficacypre","efficacypst","psolvepre","psolvepst")
9   imps$psolvepre.cgm <- imps$psolvepre - mean(imps$psolvepre)
10  imps$male.cgm <- imps$male - mean(imps$male)
11  imps$frlunch.cgm <- imps$frlunch - mean(imps$frlunch)
12  imps$teachexp.cgm <- imps$teachexp - mean(imps$teachexp)
```

The example requires the `fdir`, `lme4`, and `mitml` packages, which are loaded on lines 1 through 3. On line 4, the `set()` function of the `fdir` package identifies the file path of the folder containing the R script and sets this location as the working directory. On line 5, the `read.table` command imports the stacked data. It is only necessary to specify the name of the input data file. No file path is required when the data reside in the same folder as the R script as is the case here. Variable names are listed beginning on line 6. Importantly, the first variable named `IMPUTATION` is the index that identifies the individual files. Finally, lines 9 through 12 create new centered versions of the covariates.

The next block of code relies on the `mitml` package to fit the model to each data set and pool the results using Rubin's rules.

R Script Ex18.R, continued

```
13  implist <- as.mitml.list(split(imps, imps$imputation))
14  fit <- with(implist, lmer(psolvepst ~ psolvepre.cgm + male.cgm +
15    frlunch.cgm + teachexp.cgm + condition + (1 | school), REML = T))
16  estimates <- testEstimates(fit, extra.pars = T)
17  estimates
18  confint(estimates)
```

The `implist` command on line 13 unstacks the data and creates a list that contains the individual files. Line 14 fits the focal regression model using the `lmer` function, and line 16 uses the `testEstimates` function in `mitml` to implement Rubin's pooling rules and save the results in an object called `estimates`. Finally, lines 17 and 18 print the estimates and confidence intervals.

R Output

The table of unstandardized pooled parameter estimates is shown below. The first two columns display the pooled unstandardized estimates and standard errors, and the third through fifth columns display the corresponding test statistics. The focal model results are shown in bold typeface. The `RIV` column (relative increase in variance) is a fraction comparing imputation noise to complete-data sampling variation, and the `FMI` column (fraction of missing information in the literature) quantifies the imputation noise in each estimate as proportion of its squared standard error.

```
> estimates

Call:

testEstimates(model = fit, extra.pars = T)

Final parameter estimates and inferences obtained from 20 imputed data sets.

               Estimate Std.Error   t.value        df   P(>|t|)       RIV       FMI
(Intercept)      52.856     0.648    81.598 23181.577     0.000     0.029     0.029
psolvepre.cgm     0.466     0.034    13.897  1037.026     0.000     0.157     0.137
male.cgm          0.154     0.335     0.460   479.459     0.646     0.249     0.202
frlunch.cgm      -0.725     0.469    -1.547   220.114     0.123     0.416     0.300
teachexp.cgm      0.134     0.105     1.267  3076.432     0.205     0.085     0.079
condition         1.840     0.870     2.116 15147.975     0.034     0.037     0.036

                           Estimate
Intercept~~Intercept|school    4.462
Residual~~Residual            20.563
ICC|school                     0.178

Unadjusted hypothesis test as appropriate in larger samples.

> confint(estimates)
                    2.5 %     97.5 %
(Intercept)   51.58596983 54.1252528
psolvepre.cgm  0.40053766  0.5322427
male.cgm      -0.50500284  0.8133943
frlunch.cgm   -1.64893668  0.1987823
teachexp.cgm  -0.07312469  0.3404278
condition      0.13526144  3.5440259
```

The random intercept and within-cluster residual variances are denoted `Intercept~~Intercept|school` and `Residual~~Residual`, respectively. Moving to the coefficient section, the primary focus is the β_5 coefficient, which indicates that intervention schools scored 1.84 points higher than control schools, on average, controlling for student- and school-level covariates. The corresponding test statistic indicates that the group mean difference

is statistically different from zero (t = 2.12, p = .03). Note that these estimates are numerically identical to those from Bayesian estimation.

SECTION 5: MODELS FOR MISSING NOT AT RANDOM PROCESSES

EXAMPLE 19: MULTIPLE REGRESSION WITH A SELECTION MODEL

This example illustrates a multiple regression analysis with a selection model that invokes a missing not at random process for the outcome. The analysis uses the `behaviorachievement.dat` data set taken from a longitudinal study that followed 138 students from primary through middle school. The file includes three annual assessments of broad reading and math achievement beginning in the first grade, seventh grade standardized achievement test scores taken from a statewide assessment, and a final measure of broad reading and math obtained in ninth grade. The data also contain teacher ratings of behavioral symptoms and learning problems were also obtained in the first grade. The data description at the beginning of this document provides additional details. The variables for this analysis are as follows.

Name	Definition	Missing %	Scale
	Focal Variables		
BEHSYMP1	1st grade behavioral symptoms	3.6	Numeric
LRNPROB1	1st grade learning problems	2.2	Numeric
READ1	1st grade broad reading composite	6.5	Numeric
READ9	9th grade broad reading composite	17.4	Numeric
	Auxiliary Variables		
READ2	2nd grade broad reading composite	9.4	Numeric
STANREAD7	7th grade standardized math	19.6	Numeric
	Missing Data Indicator		
READ9MIS	9th grade reading missingness indicator	0	0 = observed, 1 = missing

Analysis Model

The analysis model features ninth grade broad reading scores regressed on first grade reading achievement and teacher-rated learning problems and behavioral symptoms.

$$READ_9 = \beta_0 + \beta_1(READ_1) + \beta_2(LRNPROB_1) + \beta_3(BEHSYMP_1) + \varepsilon \quad (34)$$

Unlike a complete-data regression analysis, all incomplete variables require distributional assumptions, including the predictors.

The missing data literature often recommends an inclusive strategy that incorporates auxiliary variables that either predict missingness or correlate with the incomplete variables (Collins et al., 2001). Following the same factored regression specification from earlier examples (e.g., Examples 2 through 6), auxiliary variables enter the model as additional outcomes that are predicted by the analysis variables and by each other. The additional regression equations are as follows.

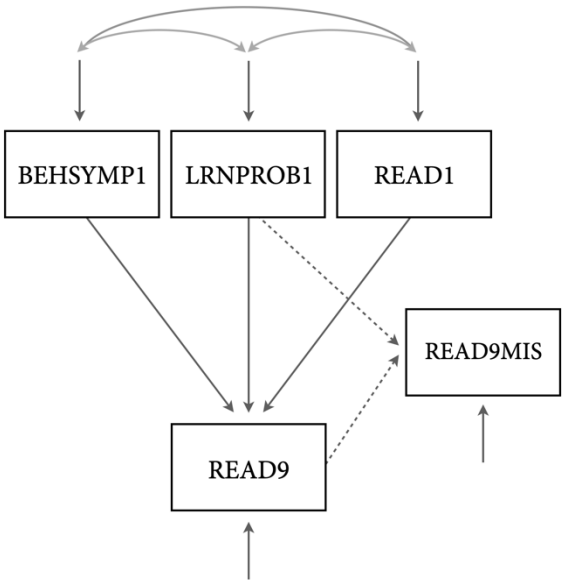
$$\begin{aligned} READ_2 &= \gamma_{01} + \gamma_{11}(READ9GRP) + \gamma_{21}(READ_1) \\ &+ \gamma_{31}(LRNPROB_1) + \gamma_{41}(BEHSYMP_1) + \epsilon_1 \\ STANREAD_7 &= \gamma_{02} + \gamma_{12}(READ_2) + \gamma_{22}(READ9GRP) \\ &+ \gamma_{32}(READ_1) + \gamma_{42}(LRNPROB_1) + \gamma_{52}(BEHSYMP_1) + \epsilon_2 \end{aligned} \tag{35}$$

Along with the focal regression model from Equation 34, the collection of regressions can be viewed as a path model, where the focal regression is one part of a larger network (see the path diagram from Example 2). The key difference is that the path coefficients are just a tool for linking incomplete variables and do not represent a substantive theory.

A missing not at random process is invoked by specifying a selection model that links the missingness probabilities to the unseen outcome scores. This model features the binary missing data indicator regressed on the outcome variable and potentially other variables. To illustrate, the missingness model also incorporates teacher-rated learning problems from first grade.

$$READ9MIS^* = \gamma_{03} + \gamma_{13}(READ_9) + \gamma_{23}(LRNPROB_1) + \epsilon_3 \tag{36}$$

The asterisk superscript denotes a normally distributed latent response variable (i.e., a probit regression). A path diagram of the focal and selection models is shown below, with dashed lines indicator the missingness model parameters. The auxiliary variable regressions follow the path diagram from Example 2.



Mplus Maximum Likelihood Estimation Script

The code block below shows Mplus script `Ex19.inp`.

Mplus Script Ex19.inp

```
DATA:
file = behaviorachievement.dat;
VARIABLE:
names = id male hispanic riskgrp atrisk behsymp1 lrnprob1
   read1 read2 read3 read9 read9grp stanread7
   math1 math2 math3 math9 math9grp stanmath7;
usevariables = read9 read1 lrnprob1 behsymp1 read2 stanread7 read9mis;
missing = all(999);
categorical = read9mis;
DATA MISSING:
names = read9;
binary = read9mis;
type = missing;
```

```
14  ANALYSIS:
15  estimator = ml;
16  link = probit;
17  integration = montecarlo;
18  MODEL:
19  read1 lrnprob1 behsymp1;
20  read9 on read1 lrnprob1 behsymp1;
21  read9mis on read9 lrnprob1;
22  read2 on read9 read1 lrnprob1 behsymp1;
23  stanread7 on read2 read9 read1 lrnprob1 behsymp1;
24  OUTPUT:
25  patterns sampstat stdyx cinterval;
```

The `DATA` command specifies the name of the input text file. No file path is required when the data set is located in the same directory as the script, as it is here. The `VARIABLE` command provides information about the data. Beginning on line 4, the `names` subcommand assigns names to the variables in the input data, the `usevariables` subcommand selects variables for the analysis, and the `missing` subcommand gives the global missing value code. Lines 10 through 13 define a binary missing data indicator called `READ9MIS`, and the preceding `categorical` subcommand on line 9 identifies the new variable as categorical.

The `DATA MISSING` command that begins on line 10 creates a binary missing data indicator. The `names` subcommand on line 11 identifies the variable to be recoded, and the `binary` command on line 12 provides a name for the new variable. Finally, the `type` subcommand on line 13 identifies the binary variable as a missing data indicator. As noted previously, the missingness indicator is identified as a categorical variable on line 9.

The `ANALYSIS` command and `estimator` subcommand specify full information maximum likelihood estimation. The default setting for a binary outcome is logistic regression. For consistency with the Bayesian analysis in Blimp, line 16 specifies a probit link that defines the binary missing data indicator as a normally distributed latent response variable. Finally, the `integration = montecarlo` subcommand invokes an algorithmic method for models with mixed variable types.

The `MODEL` section of the script consists of five lines. Listing all predictors by name on line 19 is important because doing so invokes a multivariate normal distribution for these variables. As

mentioned previously, assigning distributional assumptions to predictors is necessary for missing data handling. On line 20, the outcome variable appears to the left of the `on` keyword, and the predictors appear to the right. The missingness model from Equation 36 appears on line 21, and the two auxiliary variable regressions from Equation 35 are on lines 22 and 23. Finally, the `OUTPUT` command specifies four keywords on line 25 that request a summary of the missing data patterns, maximum likelihood estimates of sample statistics, standardized coefficients, and confidence intervals.

Mplus Output

Information about the missing data patterns is found near the top of the output file. Following the missing data pattern table, the output displays a covariance coverage matrix that gives the proportion of observed data for each variable on the diagonal and the proportion of observed data for each variable pair on the off-diagonals. The format of these table is the same as those shown in Example 1. In the interest of space, we point readers to that example for additional details.

The table of unstandardized parameter estimates is shown below. Because the analysis specifies a multivariate normal distribution for the predictors, the means, variances, and covariances of these variables are printed along with the focal model estimates. The table also reports regression models for auxiliary variables. These supporting parameters are not of substantive interest, and they do not need to be reported. The first two columns display the unstandardized estimates and their standard errors, and the third and fourth columns display the corresponding *z*-statistics and *p*-values. The focal model results are shown in bold typeface.

```
MODEL RESULTS

                                                    Two-Tailed
                  Estimate       S.E.  Est./S.E.    P-Value

 READ9      ON
    READ1             0.507      0.042     12.201      0.000
    LRNPROB1         -0.251      0.116     -2.170      0.030
    BEHSYMP1         -0.180      0.101     -1.783      0.075

READ9MIS   ON
    READ9            -0.006      0.010     -0.633      0.527
```

```
    LRNPROB1           0.042      0.013      3.150      0.002

READ2      ON
    READ9              0.676      0.065     10.373      0.000
    READ1              0.548      0.044     12.474      0.000
    LRNPROB1          -0.284      0.083     -3.428      0.001
    BEHSYMP1           0.412      0.076      5.395      0.000

 STANREAD7  ON
    READ2              1.903      0.924      2.060      0.039
    READ9              1.559      0.842      1.852      0.064
    READ1             -0.736      0.608     -1.210      0.226
    LRNPROB1           0.540      0.662      0.817      0.414
    BEHSYMP1          -0.753      0.658     -1.144      0.253

 LRNPROB1 WITH
    READ1            -11.635     19.119     -0.609      0.543

 BEHSYMP1 WITH
    READ1            -14.114     21.254     -0.664      0.507
    LRNPROB1          91.527     13.505      6.777      0.000

 Means
    READ1             86.154      1.752     49.188      0.000
    LRNPROB1          52.292      0.915     57.121      0.000
    BEHSYMP1          49.483      1.034     47.851      0.000

Intercepts
    READ9             65.832      5.832     11.287      0.000
    READ2            -19.011      5.741     -3.311      0.001
    STANREAD7         19.329     50.325      0.384      0.701

 Thresholds
    READ9MIS$1         2.715      1.305      2.080      0.038
```

```
Variances
    READ1            417.284     51.743      8.065      0.000
    LRNPROB1         114.548     13.883      8.251      0.000
    BEHSYMP1         145.486     17.587      8.272      0.000

Residual Variances
    READ9             86.368     11.474      7.528      0.000
    READ2             38.774      5.663      6.847      0.000
    STANREAD7       2206.056    303.868      7.260      0.000
```

The results are interpreted in the same way as a complete-data regression analysis. For example, consider the first-grade reading score slope. The model predicts that two individuals who differ by one point on `READ1` but are the same on `LRNPROB1` and `BEHSYMP1` should differ by 0.51 points on `READ9`. The corresponding test statistic indicates that the slope coefficient is statistically different from zero ($z = 12.20$, $p < .001$). Comparing these results to the Example 1 estimates that invoke a conditionally missing at random process provides a sensitivity check. Because the selection model estimates are virtually identical to those from Example 1, one can conclude that the regression parameters are somewhat robust to a different missingness process. This interpretation presupposes that the missingness model is correctly specified. A different set of predictors in the selection equation could change the estimates and the conclusion about robustness.

The table also reports the missingness model parameters. The outcome variable is a latent response score that represents a normally distributed propensity for missingness. To establish a metric, the latent responses are approximately scaled as a z-score. Thus, the missingness model slope coefficients essentially represent the standardized change in the missingness propensities for a one-unit increase in the predictors. The negative coefficient for `READ9` suggests that students with higher ninth grade reading scores have a lower probability of missing data in ninth grade, and the positive slope for `LRNPROB1` indicates that students with elevated learning problems in first grade are more likely to have missing data in middle school.

Blimp Bayesian Estimation Script

The code block below shows Blimp script `Ex19.inp`.

Blimp Script Ex19.imp

```
1  DATA: behaviorachievement.dat;
2  VARIABLES: id male hispanic riskgrp atrisk behsymp1 lrnprob1
3    read1 read2 read3 read9 read9grp stanread7
4    math1 math2 math3 math9 math9grp stanmath7;
5  MISSING: 999;
6  TRANSFORM:
7  read9mis = ismissing(read9);
8  ORDINAL:  read9mis;
9  MODEL:
10 focal model:
11 read9 ~ read1 lrnprob1 behsymp1;
12 missingness.model:
13 read9mis ~ read9 lrnprob1;
14 auxiliary model:
15 stanread7 read2 ~ read9 read1 lrnprob1 behsymp1;
16 SEED: 90291;
17 BURN: 1000;
18 ITERATIONS: 10000;
```

The first eight lines can be viewed as a set of commands that specify information about the data and variables. The `DATA` command specifies the name of the input text file. No file path is required when the data file is located in the same directory as the script, as it is here. Starting on line 2, the `VARIABLES` command names the data columns, and the `MISSING` command on line 5 defines a global missing value code as 999. The `TRANSFORM` command that starts on line 6 uses the `ismissing` function to create a binary missing data indicator called `READ9MIS`. The `ORDINAL` command on line 8 identifies the indicator as a binary variable.

The `MODEL` command that begins on line 9 lists the regression models, with outcome variables to the left of the tilde and predictors to the right. The focal model is listed on line 11, and the missingness (selection) model is on line 13. Line 15 is a syntax shortcut that produces the two auxiliary variable regression models in Equation 35; in the first model, `READ2` is regressed on the focal variables, and the second model features `STANREAD7` regressed on `READ2` and the focal variables. Finally, note that the `MODEL` block uses labels to order the regression summary tables on the output.

Lines 16 through 18 can be viewed as a block of commands that specify features of the MCMC algorithm: the `SEED` command gives an integer string that initializes the random number generator, the `BURN` command specifies the number of iterations for the warm-up or burn-in period, and the `ITERATIONS` command gives the number of MCMC iterations on which the analysis summaries are based (essentially, the number of MCMC cycles following the warm-up period).

Blimp Output

Prior to inspecting the parameter estimates, it is important to investigate the potential scale reduction (PSR) factor diagnostics (Gelman & Rubin, 1992) to determine whether MCMC has converged. Blimp divides the burn-in period into 20 equal segments, and it computes the PSR diagnostic for every parameter. The table located near the top of the output reports the highest (worst) PSR value across all parameters in every model. A common recommendation is that these values should be less than 1.05 or perhaps 1.10 (Asparouhov & Muthén, 2010a; Gelman et al., 2014). If the PSR in the bottom row of the table (the final check of the burn-in period) is above these cutoffs, then rerun the analysis with a longer burn-in period.

```
BURN-IN POTENTIAL SCALE REDUCTION (PSR) OUTPUT:

  NOTE: Split chain PSR is being used. This splits each chain's
        iterations to create twice as many chains.

  Comparing iterations across 2 chains     Highest PSR   Parameter #
                           26 to 50              1.273            23
                           51 to 100             1.074            40
                           76 to 150             1.081            12
                                 ...               ...            ..
                         451 to 900              1.011            14
                         476 to 950              1.007            12
                         501 to 1000             1.015            17
```

The tables summarizing the focal regression model includes unstandardized coefficients, standardized slopes, and variance explained effect size estimates. MCMC estimation produces a distribution for each parameter in the table. The median and standard deviation columns describe the center and spread of the posterior distributions; although they make no reference to drawing repeated samples, they are analogous—and numerically equivalent in most cases—to

frequentist point estimates and standard errors. The 95% credible intervals in the rightmost columns give a range that captures 95% of the parameter's distribution. These are akin to confidence intervals, but the intervals describe parameter distributions rather than characteristics of repeated samples. The `N_Eff` values in rightmost column of the table give the effective number of MCMC samples for each parameter. These quantities essentially represent the number of independent estimates on which the parameter summaries are based after removing autocorrelations from the MCMC process. Gelman et al. (2014, p. 287) recommend values greater than 100. All values in the example table exceed this recommended minimum. In cases where the `N_Eff` values are insufficient, increasing the value on the `ITERATIONS` command will remedy the issue. The table summarizing the focal regression model is shown below.

```
OUTCOME MODEL ESTIMATES:

  Summaries based on 10000 iterations using 2 chains.

  focal.model block:

Outcome Variable:  read9

Parameters                        Median     StdDev       2.5%      97.5%        PSR      N_Eff
                                 ------------------------------------------------------------------
Variances:
  Residual Var.                   91.235     12.946     70.572    120.760      1.001   6045.308

Coefficients:
  Intercept                       66.305      6.001     54.459     78.066      1.000   6237.610
  read1                            0.505      0.043      0.421      0.590      1.000   6200.956
  lrnprob1                        -0.255      0.120     -0.493     -0.017      1.000   5450.488
  behsymp1                        -0.183      0.104     -0.386      0.021      1.000   6652.942

Standardized Coefficients:
  read1                            0.687      0.040      0.599      0.755      1.000   6201.749
  lrnprob1                        -0.183      0.085     -0.346     -0.013      1.000   5371.142
  behsymp1                        -0.147      0.082     -0.306      0.017      1.000   6702.299

Proportion Variance Explained
  by Coefficients                  0.596      0.050      0.489      0.681      1.000   5971.386
  by Residual Variation            0.404      0.050      0.319      0.511      1.000   5971.386

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

The results are interpreted in the same way as a complete-data regression analysis. For example, consider the first-grade reading score slope. The model predicts that two individuals who differ by one point on `READ1` but are the same on `LRNPROB1` and `BEHSYMP1` should differ by 0.51 points on `READ9`. The 95% credible interval limits suggest this effect is statistically different

from zero ($p < .05$) because the null value is well outside the interval. Comparing these results to the Example 6 estimates that invoke a conditionally missing at random process provides a sensitivity check. Because the selection model estimates are virtually identical to those from Example 1, one can conclude that the regression parameters are somewhat robust to a different missingness process. This interpretation presupposes that the missingness model is correctly specified. A different set of predictors in the selection equation could change the estimates and the conclusion about robustness.

The table also reports the missingness model parameters. The outcome variable is a latent response score that represents a normally distributed propensity for missingness. To establish a metric, the latent responses are approximately scaled as a *z*-score. Thus, the missingness model slope coefficients essentially represent the standardized change in the missingness propensities for a one-unit increase in the predictors. The negative coefficient for `READ9` suggests that students with higher ninth grade reading scores have a lower probability of missing data in ninth grade, and the positive slope for `LRNPROB1` indicates that students with elevated learning problems in first grade are more likely to have missing data in middle school. Note that an unusually large *R*-squared value in the missingness model (e.g., greater than 70%) is often a symptom of overfitting the selection equation with too many predictors. This analysis does not exhibit that symptom.

```
  missingness.model block:

Outcome Variable:  read9mis

Parameters                        Median     StdDev       2.5%      97.5%        PSR      N_Eff
                                 ------------------------------------------------------------------
Variances:
  Residual Var.                    1.000      0.000      1.000      1.000        nan        nan

Coefficients:
  Intercept                        2.046      1.145     -0.198      4.271      1.000   2324.038
  read9                            0.011      0.009     -0.007      0.030      1.000   2053.127
  lrnprob1                        -0.038      0.012     -0.062     -0.014      1.000   2263.530

Thresholds:
  Tau 1                            0.000      0.000      0.000      0.000        nan        nan

Standardized Coefficients:

  read9                            0.146      0.119     -0.093      0.372      1.000   2089.379
  lrnprob1                        -0.362      0.104     -0.544     -0.140      1.000   2401.394
Proportion Variance Explained
  by Coefficients                  0.199      0.082      0.058      0.374      1.000   1871.736
  by Residual Variation            0.801      0.082      0.626      0.942      1.000   1871.736
```

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

Finally, the Blimp output also includes tables of regression model parameters for the auxiliary variables as well as the auto-generated models for incomplete predictors. These additionally results are not of substantive interest and would not be reported. The auxiliary variable models appear in `OUTCOME MODEL ESTIMATES` section with the focal results, and the auto-generated predictor models are displayed under the heading `PREDICTOR MODEL ESTIMATES`.

EXAMPLE 20: MULTIPLE REGRESSION WITH A PATTERN MIXTURE MODEL

This example illustrates a pattern mixture regression model that invokes a missing not at random process for the outcome. The analysis uses the `behaviorachievement.dat` data set taken from a longitudinal study that followed 138 students from primary through middle school. The file includes three annual assessments of broad reading and math achievement beginning in the first grade, seventh grade standardized achievement test scores taken from a statewide assessment, and a final measure of broad reading and math obtained in ninth grade. The data also contain teacher ratings of behavioral symptoms and learning problems were also obtained in the first grade. The data description at the beginning of this document provides additional details. The variables for this analysis are as follows.

Name	Definition	Missing %	Scale
	Focal Variables		
BEHSYMP1	1st grade behavioral symptoms	3.6	Numeric
LRNPROB1	1st grade learning problems	2.2	Numeric
READ1	1st grade broad reading composite	6.5	Numeric
READ9	9th grade broad reading composite	17.4	Numeric
	Auxiliary Variables		
READ2	2nd grade broad reading composite	9.4	Numeric
STANREAD7	7th grade standardized math	19.6	Numeric
	Missing Data Indicator		
READ9MIS	9th grade reading missingness indicator	0	0 = observed, 1 = missing

Analysis Model

The population-level analysis model features ninth grade broad reading scores regressed on first grade reading achievement and teacher-rated learning problems and behavioral symptoms.

$$READ_9 = \beta_0 + \beta_1(READ_1) + \beta_2(LRNPROB_1) + \beta_3(BEHSYMP_1^*) + \varepsilon \quad (37)$$

Unlike a complete-data regression analysis, all incomplete variables require distributional assumptions, including the predictors.

A missing not at random process is invoked by specifying a pattern mixture model that links the missingness probabilities to the unseen outcome scores. This model features the binary missing data indicator as a predictor and possibly a moderator. The basic idea is that the missing data patterns define subgroups with different parameter values. This example illustrates a process where students with missing scores in ninth grade have a lower reading mean. It is also possible for the regression coefficients to differ by pattern (see Enders, 2022, Section 9.8).

To invoke a missing data pattern-specific mean difference, the fitted model includes the binary missing data indicator as a predictor

$$\begin{aligned} READ_9 = \beta_{0(\text{com})} + \beta_{0(\text{diff})}(READ9MIS) + \beta_1(READ_1) \\ + \beta_2(LRNPROB_1) + \beta_3(BEHSYMP_1) + \varepsilon \end{aligned} \tag{38}$$

such that $\beta_{0(\text{com})}$ is the intercept (mean level) for students with complete reading scores, and $\beta_{0(\text{diff})}$ is outcome mean difference for students with missing data. The intercept coefficient from Equation 37 is a weighted average of the group-specific intercepts

$$\beta_0 = p_{(\text{com})}\beta_{0(\text{com})} + p_{(\text{mis})}\left(\beta_{0(\text{com})} + \beta_{0(\text{diff})}\right) = p_{(\text{com})}\beta_{0(\text{com})} + p_{(\text{mis})}\beta_{0(\text{mis})} \tag{39}$$

where $p_{(\text{com})}$ and $p_{(\text{mis})}$ are the proportions of complete and missing outcome scores, respectively. Importantly, $\beta_{0(\text{diff})}$ is not estimable from the data, and researchers must provide a value that induces the posited missing not at random process (e.g., students with missing outcome data have lower reading levels). Following the procedure described in Enders (2022), the scripts below set $\beta_{0(\text{diff})}$ to a value that is 0.20 standard deviation units below $\beta_{0(\text{com})}$. That is, the average reading level for students with missing outcome scores is lower by an amount commensurate with Cohen's (1988) small effect size benchmark.

Unlike a complete-data regression analysis, incomplete predictor variables also require distributional assumptions and models that define those distributions. The analysis uses a factored regression specification that uses a sequence of univariate regression models to link incomplete predictors. This specification was introduced in Examples 2 through 5. The additional regression equations are as follows.

$$READ9MIS^* = \gamma_{01} + \epsilon_1 \tag{40}$$

$$BEHSYMP_1^* = \gamma_{02} + \gamma_{12}(READ9MIS) + \epsilon_2$$

$$LRNPROB_1 = \gamma_{03} + \gamma_{13}(BEHSYMP_1^*) + \gamma_{23}(READ9MIS) + \epsilon_3$$

$$READ_1 = \gamma_{04} + \gamma_{14}(LRNPROB_1) + \gamma_{24}(BEHSYMP_1^*) + \gamma_{34}(READ9MIS) + \epsilon_4$$

The asterisk subscript in the `READ9MIS` model denotes a latent response variable (i.e., probit regression). Listing the missing data indicator first in the sequence is important because pattern proportions needed for Equation 39 are a function of the empty model's regression intercept.

The missing data literature often recommends an inclusive strategy that incorporates auxiliary variables that either predict missingness or correlate with the incomplete variables (Collins et al., 2001). Following the same factored regression specification from earlier examples (e.g., Examples 2 through 6), auxiliary variables enter the model as additional outcomes that are predicted by the analysis variables and by each other. The additional regression equations are as follows.

$$\begin{aligned} READ_2 &= \gamma_{05} + \gamma_{15}(READ_9) + \gamma_{25}(READ_1) \\ &+ \gamma_{35}(LRNPROB_1) + \gamma_{45}(BEHSYMP_1^*) + \gamma_{55}(READ9MIS) + \epsilon_5 \\ STANREAD_7 &= \gamma_{06} + \gamma_{16}(READ_2) + \gamma_{26}(READ_9) + \gamma_{36}(READ_1) \\ &+ \gamma_{46}(LRNPROB_1) + \gamma_{56}(BEHSYMP_1^*) + \gamma_{66}(READ9MIS) + \epsilon_6 \end{aligned} \tag{41}$$

Along with the focal regression from Equation 37 and the predictor models from Equation 40, the collection of regressions can be viewed as a path model, where the focal regression is one part of a larger network (see the path diagram from Example 2). The key difference is that the path coefficients are just a tool for linking incomplete variables and do not represent a substantive theory.

Mplus Maximum Likelihood Estimation Script

The code block below shows Mplus script `Ex20.inp`.

Mplus Script Ex20.inp

```
1   DATA:
```

```
file = behaviorachievement.dat;
VARIABLE:
names = id male hispanic riskgrp atrisk behsymp1 lrnprob1
  read1 read2 read3 read9 read9grp stanread7
  math1 math2 math3 math9 math9grp stanmath7;
usevariables = read9 read1 lrnprob1 behsymp1 read2 stanread7 read9mis;
missing = all(999);
categorical = read9mis;
DATA MISSING:
names = read9;
binary = read9mis;
type = missing;
ANALYSIS:
estimator = ml;
link = probit;
integration = montecarlo;
MODEL:
[read9mis$1] (missmean);
behsymp1 on read9mis;
lrnprob1 on behsymp1 read9mis;
read1 on lrnprob1 behsymp1 read9mis;
read9 on read9mis (beta0diff)
   read1 lrnprob1 behsymp1;
[read9] (beta0com); read9 (resvar);
read2 on read9 read1 lrnprob1 behsymp1 read9mis;
stanread7 on read2 read9 read1 lrnprob1 behsymp1 read9mis;
MODEL CONSTRAINT:
new(cohensd pcom pmis beta0);

cohensd = -.20;
beta0diff = cohensd * sqrt(resvar);
pmis = phi(-missmean);
pcom = 1 - pmis;
beta0 = (beta0com * pcom) + ((beta0com + beta0diff) * pmis);
OUTPUT:
patterns sampstat stdyx cinterval;
```

The `DATA` command specifies the name of the input text file. No file path is required when the data set is located in the same directory as the script, as it is here. The `VARIABLE` command

provides information about the data. Beginning on line 4, the `names` subcommand assigns names to the variables in the input data, the `usevariables` subcommand selects variables for the analysis, and the `missing` subcommand gives the global missing value code. Lines 10 through 13 define a binary missing data indicator called `READ9MIS`, and the preceding `categorical` subcommand on line 9 identifies the new variable as categorical.

The `DATA MISSING` command that begins on line 10 creates a binary missing data indicator. The `names` subcommand on line 11 identifies the variable to be recoded, and the `binary` command on line 12 provides a name for the new variable. Finally, the `type` subcommand on line 13 identifies the binary variable as a missing data indicator. As noted previously, the missingness indicator is identified as a categorical variable on line 9.

The `ANALYSIS` command and `estimator` subcommand specify full information maximum likelihood estimation. The default setting for a binary outcome is logistic regression. For consistency with the Bayesian analysis in Blimp, line 16 specifies a probit link that defines the binary missing data indicator as a normally distributed latent response variable. Finally, the `integration = montecarlo` subcommand invokes an algorithmic method for models with mixed variable types.

The `MODEL` command that begins on line 18 lists the regression models, with outcome variables to the left of the `on` keyword and predictors to the right. An empty model for the missing data indicator is listed on line 19. The label on the threshold parameter from this model (`missmean`) is used later in the code to compute the missing data pattern proportions. The remaining predictor models from Equation 40 appear on lines 20 through 22. Next, lines 23 through 25 list the focal model parameters. Line 23 assigns a label to the pattern mean difference (i.e., the $\beta_{0(\text{diff})}$ coefficient from Equation 38), and line 25 labels the complete-case intercept and residual variance, respectively. Collectively, the labels are used later in the code to induce the desired effect size difference for the missing scores. Finally, lines 26 and 27 produce the two auxiliary variable regression models from Equation 41; in the first model, `READ2` is regressed on the focal variables, and the second model features `STANREAD7` regressed on `READ2` and the focal variables.

The `MODEL CONSTRAINT` section of the script from lines 28 through 34 includes commands that define new parameters and impose constraints. First, line 29 assigns names to four new parameters. Line 30 provides the desired effect size difference for the group with missing data, and line 31 defines a mean difference parameter `beta0diff` that is a function of the effect size and residual standard deviation (see Enders, 2022, Eq. 9.29). Lines 32 and 33 use the threshold

parameter from the missing data indicator's model to compute the missing data pattern proportions. Line 34 computes the weighted intercept that averages over the missing data patterns (see Equation 39). Finally, the `OUTPUT` command specifies four keywords on line 36 that request a summary of the missing data patterns, maximum likelihood estimates of sample statistics, standardized coefficients, and confidence intervals.

Mplus Output

Information about the missing data patterns is found near the top of the output file. Following the missing data pattern table, the output displays a covariance coverage matrix that gives the proportion of observed data for each variable on the diagonal and the proportion of observed data for each variable pair on the off-diagonals. The format of these table is the same as those shown in Example 1. In the interest of space, we point readers to that example for additional details.

The table of unstandardized parameter estimates is shown below. The table reports regression models for predictor variables and auxiliary variables. These supporting parameters are not of substantive interest, and they do not need to be reported. The first two columns display the unstandardized estimates and their standard errors, and the third and fourth columns display the corresponding *z*-statistics and *p*-values. The focal model results are shown in bold typeface.

```
MODEL RESULTS

                                                    Two-Tailed
                  Estimate       S.E.  Est./S.E.    P-Value

 BEHSYMP1   ON
    READ9MIS          7.692      2.654      2.898      0.004

 LRNPROB1   ON
    BEHSYMP1          0.597      0.054     10.967      0.000
    READ9MIS          4.241      1.728      2.454      0.014

READ1      ON
    LRNPROB1         -0.012      0.244     -0.048      0.961
    BEHSYMP1         -0.064      0.208     -0.306      0.760
    READ9MIS         -3.230      4.861     -0.664      0.506
```

READ9 ON				
READ9MIS	-1.862	0.124	-14.959	0.000
READ1	**0.504**	**0.042**	**11.990**	**0.000**
LRNPROB1	**-0.248**	**0.117**	**-2.108**	**0.035**
BEHSYMP1	**-0.181**	**0.101**	**-1.790**	**0.073**
READ2 ON				
READ9	0.674	0.065	10.344	0.000
READ1	0.551	0.044	12.563	0.000
LRNPROB1	-0.290	0.084	-3.440	0.001
BEHSYMP1	0.414	0.077	5.415	0.000
READ9MIS	1.124	2.244	0.501	0.617
STANREAD7 ON				
READ2	1.891	0.923	2.048	0.041
READ9	1.590	0.841	1.891	0.059
READ1	-0.733	0.609	-1.205	0.228
LRNPROB1	0.493	0.678	0.728	0.467
BEHSYMP1	-0.737	0.659	-1.118	0.264
READ9MIS	6.119	13.186	0.464	0.643
Intercepts				
READ9	66.040	5.887	11.218	0.000
READ1	90.485	9.209	9.826	0.000
LRNPROB1	21.999	2.715	8.103	0.000
BEHSYMP1	48.148	1.104	43.610	0.000
READ2	-19.003	5.776	-3.290	0.001
STANREAD7	18.400	50.557	0.364	0.716
Thresholds				
READ9MIS$1	0.939	0.126	7.472	0.000
Residual Variances				
READ9	**86.643**	**11.584**	**7.480**	**0.000**
READ1	414.570	51.322	8.078	0.000
LRNPROB1	54.545	6.727	8.109	0.000
BEHSYMP1	137.009	16.565	8.271	0.000
READ2	38.759	5.658	6.850	0.000
STANREAD7	2200.450	303.283	7.255	0.000

```
New/Additional Parameters
    COHENSD          -0.200      0.000  *********      0.000
    PCOM              0.826      0.032     25.608      0.000
    PMIS              0.174      0.032      5.389      0.000
    BETA0            65.716      5.887     11.162      0.000
```

The regression slopes interpreted in the same way as a complete-data regression analysis. For example, consider the first-grade reading score slope. The model predicts that two individuals who differ by one point on `READ1` but are the same on `LRNPROB1` and `BEHSYMP1` should differ by 0.51 points on `READ9`. The corresponding test statistic indicates that the slope coefficient is statistically different from zero ($z = 11.99$, $p < .001$). The `read9mis` coefficient from the same table is the pattern mean difference $\beta_{0(\text{diff})}$ (see Equation 38). The `MODEL CONSTRAINT` command defined a set of new model parameters, including weighted average intercept. The table summarizing the additional parameters is shown below. These quantities are found under the table labeled `New/Additional Parameters`. The weighted intercept coefficient that averages over the missing data patterns is labeled `beta0`.

Comparing these results to the Example 1 estimates that invoke a conditionally missing at random process provides a sensitivity check that conveys the impact of a missing not at random process where students with missing data have lower mean reading levels in ninth grade. This comparison presupposes that the missingness model is correctly specified. The missing data indicator could also moderate associations in the regression model, in which case the estimates and conclusions about robustness could change.

Blimp Bayesian Estimation Script

The code block below shows Blimp script `Ex20.inp`.

Blimp Script Ex20.imp

```
1  DATA: behaviorachievement.dat;
2  VARIABLES: id male hispanic riskgrp atrisk behsymp1 lrnprob1
3    read1 read2 read3 read9 read9grp stanread7
4    math1 math2 math3 math9 math9grp stanmath7;
5  MISSING: 999;
```

```
6   TRANSFORM:
7   read9mis = ismissing(read9);
8   ORDINAL:  read9mis;
9   MODEL:
10  focal model:
11  read9 ~ 1@beta0com read9mis@beta0diff read1 lrnprob1 behsymp1;
12  read9 ~~ read9@resvar;
13  indicator.model:
14  read9mis ~ 1@missmean;
15  predictor.model:
16  read1 lrnprob1 behsymp1 ~ read9mis;
17  auxiliary model:
18  stanread7 read2 ~ read9 read1 lrnprob1 behsymp1;
19  PARAMETERS:
20  cohensd = -.20;
21  beta0diff = cohensd * sqrt(resvar);
22  pmis = phi(missmean);
23  pcom = 1 - pmis;
24  beta0 = (beta0com * pcom) + ((beta0com + beta0diff) * pmis);
25  SEED: 90291;
26  BURN: 1000;
27  ITERATIONS: 10000;
```

The first eight lines can be viewed as a set of commands that specify information about the data and variables. The `DATA` command specifies the name of the input text file. No file path is required when the data file is located in the same directory as the script, as it is here. Starting on line 2, the `VARIABLES` command names the data columns, and the `MISSING` command on line 5 defines a global missing value code as 999. The `TRANSFORM` command that starts on line 6 uses the `ismissing` function to create a binary missing data indicator called `READ9MIS`. The `ORDINAL` command on line 8 identifies the indicator as a binary variable.

The `MODEL` command that begins on line 9 lists the regression models, with outcome variables to the left of the tilde and predictors to the right. The code uses model block labels (`focal.model`, `indicator.model`, `predictor.model`, and `auxiliary.model`) to group the regressions and order output tables. The focal model listed on line 11 assigns labels to intercept and the pattern mean difference (i.e., the $\beta_{0(\text{com})}$ and $\beta_{0(\text{diff})}$ coefficients from Equation 38) using the @ symbol. The residual variance is also labeled on line 12. Collectively, the labels are used later in the code to induce the desired effect size difference for the missing scores. An empty

model for the missing data indicator is listed on line 14. The label on the intercept parameter is used later in the code to compute the missing data pattern proportions. Line 16 is a syntax shortcut that produces the predictor regression models in Equation 40; in the first model, `BEHSYMP1` is regressed on the binary missing data indicator `READ9MIS`, the second model features `LRNPROB1` regressed on `BEHSYMP1` and the indicator, and the third regression features `READ1` regressed on all other predictors. Line 18 is a similar syntax shortcut that produces the two auxiliary variable regression models in Equation 41; in the first model, `READ2` is regressed on the focal variables, and the second model features `STANREAD7` regressed on `READ2` and the focal variables.

The `PARAMETERS` section of the script from lines 19 through 25 includes commands that define new parameters and impose constraints. Line 20 provides the desired effect size difference for the group with missing data, and line 21 defines a mean difference parameter `beta0diff` that is a function of the effect size and residual standard deviation (see Enders, 2022, Eq. 9.29). Lines 22 and 23 use the intercept parameter from the missing data indicator's model to compute the missing data pattern proportions. Finally, line 24 computes the weighted intercept that averages over the missing data patterns (see Equation 39).

Lines 25 through 27 can be viewed as a block of commands that specify features of the MCMC algorithm: the `SEED` command gives an integer string that initializes the random number generator, the `BURN` command specifies the number of iterations for the warm-up or burn-in period, and the `ITERATIONS` command gives the number of MCMC iterations on which the analysis summaries are based (essentially, the number of MCMC cycles following the warm-up period).

Blimp Output

Prior to inspecting the parameter estimates, it is important to investigate the potential scale reduction (PSR) factor diagnostics (Gelman & Rubin, 1992) to determine whether MCMC has converged. Blimp divides the burn-in period into 20 equal segments, and it computes the PSR diagnostic for every parameter. The table located near the top of the output reports the highest (worst) PSR value across all parameters in every model. A common recommendation is that these values should be less than 1.05 or perhaps 1.10 (Asparouhov & Muthén, 2010a; Gelman et al., 2014). If the PSR in the bottom row of the table (the final check of the burn-in period) is above these cutoffs, then rerun the analysis with a longer burn-in period.

```
BURN-IN POTENTIAL SCALE REDUCTION (PSR) OUTPUT:

  NOTE: Split chain PSR is being used. This splits each chain's
        iterations to create twice as many chains.

  Comparing iterations across 2 chains     Highest PSR   Parameter #
                           26 to 50              1.200            48
                           51 to 100             1.118            60
                           76 to 150             1.190            45
                                 ...               ...            ..
                          451 to 900             1.010            58
                          476 to 950             1.026            50
                          501 to 1000            1.014            50
```

The tables summarizing the focal regression model includes unstandardized coefficients, standardized slopes, and variance explained effect size estimate. MCMC estimation produces a distribution for each parameter in the table. The median and standard deviation columns describe the center and spread of the posterior distributions; although they make no reference to drawing repeated samples, they are analogous—and numerically equivalent in most cases—to frequentist point estimates and standard errors. The 95% credible intervals in the rightmost columns give a range that captures 95% of the parameter's distribution. These are akin to confidence intervals, but the intervals describe parameter distributions rather than characteristics of repeated samples. The `N_Eff` values in rightmost column of the table give the effective number of MCMC samples for each parameter. These quantities essentially represent the number of independent estimates on which the parameter summaries are based after removing autocorrelations from the MCMC process. Gelman et al. (2014, p. 287) recommend values greater than 100. All values in the example table exceed this recommended minimum. In cases where the `N_Eff` values are insufficient, increasing the value on the `ITERATIONS` command will remedy the issue. The table summarizing the focal regression model is shown below.

```
OUTCOME MODEL ESTIMATES:

  Summaries based on 10000 iterations using 2 chains.

  focal.model block:

Outcome Variable:  read9

Parameters                          Median     StdDev       2.5%      97.5%        PSR      N_Eff
                                   ------------------------------------------------------------------
```

```
Variances:
  Residual Var.                    91.981     12.979     70.576    120.994      1.000   6177.696

Coefficients:
  Intercept                        65.956      6.068     54.052     77.845      1.000   6431.847
  read9mis                         -1.918      0.133     -2.200     -1.680      1.000   6164.704
  read1                             0.505      0.043      0.419      0.589      1.001   6801.698
  lrnprob1                         -0.248      0.122     -0.488     -0.006      1.000   5902.838
  behsymp1                         -0.180      0.104     -0.385      0.026      1.000   7368.217

Standardized Coefficients:
  read9mis                         -0.048      0.004     -0.057     -0.041      1.000  11312.195
  read1                             0.684      0.040      0.597      0.750      1.000   6796.046
  lrnprob1                         -0.176      0.085     -0.344     -0.004      1.000   5997.788
  behsymp1                         -0.144      0.083     -0.304      0.020      1.000   7364.186

Proportion Variance Explained
  by Coefficients                   0.600      0.049      0.494      0.684      1.000   6470.849
  by Residual Variation             0.400      0.049      0.316      0.506      1.000   6470.849

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

The regression slopes interpreted in the same way as a complete-data regression analysis. For example, consider the first-grade reading score slope. The model predicts that two individuals who differ by one point on `READ1` but are the same on `LRNPROB1` and `BEHSYMP1` should differ by 0.51 points on `READ9`. The 95% credible interval limits suggest this effect is statistically different from zero ($p < .05$) because the null value is well outside the interval. This table does not display the regression intercept. Rather, `Intercept` and `read9mis` coefficients are the pattern-specific parameters, $\beta_{0(\text{com})}$ and $\beta_{0(\text{diff})}$ (see Equation 38).

The `PARAMETERS` command defined a set of new model parameters, including weighted average intercept. The table summarizing the additional parameters is shown below.

```
GENERATED PARAMETERS:

  Summaries based on 10000 iterations using 2 chains.


Parameters                         Median     StdDev       2.5%      97.5%        PSR      N_Eff
                                 ----------------------------------------------------------------

  cohensd                          -0.200      0.000     -0.200     -0.200      1.000      2.001
  beta0diff                        -1.918      0.133     -2.200     -1.680      1.000   6165.475
  pmis                              0.174      0.032      0.117      0.243      1.000   2800.948
  pcom                              0.826      0.032      0.757      0.883      1.000   2800.948
  beta0                            65.611      6.068     53.728     77.519      1.000   6424.868
```

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

The weighted intercept coefficient that averages over the missing data patterns is labeled `beta0`. Comparing these results to the Example 6 estimates that invoke a conditionally missing at random process provides a sensitivity check that conveys the impact of a missing not at random process where students with missing data have lower mean reading levels in ninth grade. This comparison presupposes that the missingness model is correctly specified. The missing data indicator could also moderate associations in the regression model, in which case the estimates and conclusions about robustness could change. Finally, the Blimp output also includes summary tables for the predictor and auxiliary variable models. These additionally results are not of substantive interest and would not be reported.

REFERENCES

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

Asparouhov, T., & Muthén, B. (2010b). *Chi-square statistics with multiple imputation*. Retrieved 2/4/2016, from https://www.statmodel.com/download/MI7.pdf

Asparouhov, T., & Muthén, B. (2021). Advances in Bayesian model fit evaluation for structural equation models. *Structural Equation Modeling: A Multidisciplinary Journal*, *28*, 1–14.

Barnard, J., & Rubin, D. B. (1999). Small-sample degrees of freedom with multiple imputation. *Biometrika*, *86*, 948–955. https://doi.org/DOI 10.1093/biomet/86.4.948

Cohen, J. (1988). *Statistical power analysis for the behavioral sciences* (2nd ed.). Erlbaum.

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

Enders, C. K. (2022). *Applied Missing Data Analysis* (2nd ed.). Guilford Press.

Gelman, A., et al. (2014). *Bayesian data analysis* (3rd ed.). CRC Press.

Gelman, A., & Rubin, D. B. (1992). Inference from iterative simulation using multiple sequences. *Statistical Science*, *7*, 457–472. https://doi.org/10.1214/ss/1177011136

Graham, J. W. (2003). Adding missing-data-relevant variables to FIML-based structural equation models. *Structural Equation Modeling: A Multidisciplinary Journal*, *10*, 80–100. https://doi.org/10.1207/S15328007sem1001_4

Li, K. H., et al. (1991). Large-sample significance levels from multiply imputed data using moment-based statistics and an F reference distribution. *Journal of the American Statistical Association*, *86*, 1065–1073. https://doi.org/Doi 10.2307/2290525

Little, R. J. A., & Rubin, D. B. (2020). *Statistical analysis with missing data* (3rd ed.). Wiley.

Montague, M., et al. (2005). Academic and behavioral outcomes for students at risk for emotional and behavioral disorders. *Behavioral Disorders*, *31*, 84–94.

Montague, M., et al. (2014). The effects of cognitive strategy instruction on math problem solving of middle-school students of varying ability. *Journal of Educational Psychology*, *106*, 469–481.

Rights, J. D., & Sterba, S. K. (2019). Quantifying explained variance in multilevel models: An integrative framework for defining R-squared measures. *Psychological Methods*, *24*, 309–338. https://doi.org/dx.doi.org/10.1037/met0000184