FSEM Analysis Examples

Fitting Structural Equation Models with Latent Variables

Brian T. Keller and Craig K. Enders

Contents

Required Packages

This R script requires the following packages:

1. `rblimp`
2. `remotes` (required to install below)

```
# Install `rblimp` package from Github (Uncomment line below to install)
# remotes::install_github('blimp-stats/rblimp')

# Load required package
library(rblimp)
```

Reading Data

Load data sets from CSV files hosted on GitHub. The data sets are read directly from the online repository and will be used for various structural equation modeling examples throughout this script.

```
# Base url for Github Repository
baseurl <- 'https://raw.githubusercontent.com/blimp-stats/fsem/refs/heads/main/data'

# Read in all data sets for examples
data1 <- read.csv(paste0(baseurl, '/data1.csv'))
```

```r
data2 <- read.csv(paste0(baseurl, '/data2.csv'))
data3 <- read.csv(paste0(baseurl, '/data3.csv'))
data4 <- read.csv(paste0(baseurl, '/data4.csv'))
data5 <- read.csv(paste0(baseurl, '/data5.csv'))
```

General Overview of `rblimp` Package

The `rblimp` package is an R interface for Blimp. It allows one to interact with Blimp using R's interface, to leverage the power of R's ecosystem.

The main function is `rblimp()` which will create the required input file, run Blimp, and load the resulting output file into R, including any saved data, estimates, and output.

To illustrate, let's fit a simple one predictor model.

```r
# Fitting a simple one predictor model in `rblimp`
model0 <- rblimp(
    data = data1,
    model = 'y ~ x1',
    seed = 90291,
    burn = 10000,
    iter = 10000
)
```

The above model syntax can be read as `y` predicted (`~`) by `x1`. By default, for manifest variables Blimp will include an intercept. Note, below often we will use a colon between two variables (e.g., x1:x4). This denotes a range, such as x1 to x4.

After fitting the model, output can be provided in one of the following formats.

- Raw Blimp output `output(model0)`
- R summary of model estimates `summary(model0)`
- HTML APA-like table of model estimates `model_table(model0)`
- Posterior plots `posterior_plot(model0)`

While the Raw output provides one with complete information, including sample statistics, missing data patterns, fit information, the others provide a quick summary of the estimates.

```r
# Raw Blimp Output
output(model0)

# R Summary of estimates
summary(model0)

# HTML Table of estimates
model_table(model0)

# Plots of posterior distributions
posterior_plot(model0)
```

Models 1-3: Basic Latent Variable Models

The first three models demonstrate basic latent variable modeling with normal outcomes. These models introduce foundational concepts including:

- Latent predictor variables

- Normal factor indicators
- Mediation with latent variables
- Correlated latent and manifest predictors
- Correlated indicator residuals

All these models use normal outcome variables and provide the foundation for more complex models later in this script.

MODEL 1: Normal Outcome with Latent Predictor

Model Specification

- Normal outcome variable
- Latent predictor (eta)
- Normal factor indicators (x1:x4)

Below specifies the model in `rblimp`.

Model 1a: Basic specification

```
# Fit latent variable model using MCMC estimation
model1a <- rblimp(
    data = data1,
    latent = 'eta',
    model = '
        eta -> x1:x4;
        y ~ eta;',
    seed = 90291,
    burn = 10000,
    iter = 10000
)
output(model1a)
```

Blimp requires users to specify the name of the latent variable. The `latent` command above, states that we are creating a new unobserved variable called `eta`. The `->` is used to specify the variable on the left (e.g., `eta`) predicting the variables on the right (e.g., `x1`, `x2`, `x3`, `x4`). When using the `->` with a latent variable on the left hand side, by default Blimp will fix the first loading to 1 for identification (so called marker variable scaling). By default, latent intercepts (means) are set to 0, manifest intercepts are estimated, and all variances/residual variances are estimated.

Below, we can specify the fixed factor scaling, where we fix the variance of the latent variable (`eta`) to 1. The first line of the model below, specifies this (`eta ~~ eta@1`). The next line provides a label for the loading of `x1` so that it is not fixed to 1.

Model 1b: Alternative specification with constraints

```
# Fix latent variance to 1 and set loading constraint
model1b <- rblimp(
    data = data1,
    latent = 'eta',
    model = '
        eta ~~ eta@1;
        eta -> x1@lo1 x2:x4;
        y ~ eta;',
    seed = 90291,
    burn = 10000,
```

```
    iter = 10000
)
output(model1b)
```

MODEL 2: Mediation Model with Latent Mediator

Model Specification

- Normal outcome variable
- Normal manifest predictors (m1, m2)
- Latent mediator (eta)
- Normal factor indicators (x1:x4)
- Correlated indicator residuals

Below, we have incorporated two predictors to create the latent mediation model. By default, Blimp will not estimate an intercept for the latent variable (unless explicitly specified via adding `intercept` or `1` to the right hand side of the model specification).

```
# Fit mediation model with latent mediator
model2 <- rblimp(
    data = data1,
    latent = 'eta',
    model = '
        eta ~ m1 m2;
        eta ~~ eta@1;
        eta -> x1@lo1 x2:x4;
        y ~ eta m1 m2;',
    seed = 90291,
    burn = 10000,
    iter = 10000
)
output(model2)
```

MODEL 3: Correlated Predictors Model

Model Specification

- Normal outcome variable
- Correlated latent and normal manifest predictors
- Normal factor indicators (x1:x4)
- Correlated indicator residuals (x3, x4)

In this example, we illustrate how to specify correlations between two predictors. Below, we specify that the latent `eta` is correlated with the manifest `m1` via `~~`. This allows us to have an unspecified direction to the relationship. Similarly, we can specify the correlated residuals of two items with the same syntax.

```
# Fit model with correlated latent and manifest predictors
model3 <- rblimp(
    data = data1,
    latent = 'eta',
    model = '
        eta ~~ m1;
        eta ~~ eta@1;
        eta -> x1@lo1 x2:x4;
        x3 ~~ x4;
        y ~ eta m1;',
```

```
    seed = 90291,
    burn = 10000,
    iter = 10000
)
output(model3)
```

Models 4-8: Ordinal Indicators and Various Outcome Types

Models 4-8 extend the basic latent variable framework to accommodate different types of variables. These models demonstrate key concepts including:

- Ordinal and binary factor indicators
- Latent response variables for categorical predictors
- Multicategorical (nominal) predictors
- Various outcome types: normal, binary, multicategorical, and count

These models build on the foundation from Models 1-3 and introduce techniques for handling non-normal data in structural equation models.

MODEL 4: Model with Ordinal Indicators

Model Specification

- Normal outcome variable
- Correlated latent and binary manifest predictors
- Ordinal factor indicators (xc1:xc4)
- Correlated indicator residuals (xc3, xc4)

In this example, we illustrate how to specify ordinal factor indicators. The `ordinal` input below indicates that `m_bin` and `xc1` to `xc4` are ordinal data. Blimp uses a probit model to account for the ordinal nature of the data. The first threshold is fixed to 0 for identification, and C – 1 thresholds are estimated, where C is the number of response categories. In addition, Blimp allows for correlations to be estimated for the ordinal data with the underlying latent propensity in the probit model.

```
# Fit model with ordinal factor indicators
model4 <- rblimp(
    data = data1,
    ordinal = 'm_bin xc1:xc4',
    latent = 'eta',
    model = '
        eta ~~ m_bin;
        eta ~~ eta@1;
        eta -> xc1@lo1 xc2:xc4;
        xc3 ~~ xc4;
        y ~ eta m_bin;',
    seed = 90291,
    burn = 20000,
    iter = 20000
)
output(model4)
```

MODEL 5: Latent Response Variable Model

Model Specification:

- Normal outcome variable

- Correlated latent and binary manifest predictors
- Latent response variable replaces binary predictor
- Ordinal factor indicators (xc1:xc4)
- Correlated indicator residuals (xc3, xc4)

In this example, we illustrate how to use the latent response variable for a binary predictor. This is similar to Model 4, but instead of using the observed binary predictor `m_bin`, we use the underlying latent propensity. By adding `.latent` to the end of the variable name, Blimp will use the latent response variable instead of the observed variable. This type of specification is analogous to other SEM software when using probit models.

```
# Fit model using latent response variable for binary predictor
model5 <- rblimp(
    data = data1,
    ordinal = 'm_bin xc1:xc4',
    latent = 'eta',
    model = '
        eta ~~ m_bin;
        eta ~~ eta@1;
        eta -> xc1@lo1 xc2:xc4;
        xc3 ~~ xc4;
        y ~ eta m_bin.latent;',
    seed = 90291,
    burn = 20000,
    iter = 20000
)
output(model5)
```

MODEL 6: Multicategorical Predictor Model

Model Specification

- Normal outcome variable
- Multicategorical exogenous predictor (m_nom)
- Ordinal factor indicators (xc1:xc4)
- Correlated indicator residuals (xc3, xc4)

In this example, we illustrate how to specify multicategorical (nominal) predictors. The `nominal` input below indicates that `m_nom` is a nominal variable. Blimp uses dummy coding for nominal variables, where the lowest numeric code is the reference group. For a variable with K categories, K-1 dummy variables are created automatically. By using the nominal variable as a predictor, the dummy codes are used instead.

```
# Fit model with multicategorical nominal predictor
model6 <- rblimp(
    data = data1,
    ordinal = 'xc1:xc4',
    nominal = 'm_nom',
    latent = 'eta',
    model = '
        eta ~ m_nom;
        eta ~~ eta@1;
        eta -> xc1@lo1 xc2:xc4;
        xc3 ~~ xc4;
        y ~ eta m_nom;',
    seed = 90291,
    burn = 20000,
    iter = 20000
```

```
)
output(model6)
```

MODEL 7A: Binary Outcome Model

Model Specification

- Binary outcome variable (y_bin)
- Multicategorical exogenous predictor (m_nom)
- Ordinal factor indicators (xc1:xc4)
- Correlated indicator residuals (xc3, xc4)

In this example, we illustrate how to specify a binary outcome variable. By specifying `y_bin` in the `nominal` input, Blimp will recode the lowest numeric code is the reference group. A logistic regression model is used.

```
# Fit model with binary outcome
model7a <- rblimp(
    data = data1,
    ordinal = 'xc1:xc4',
    nominal = 'm_nom y_bin',
    latent = 'eta',
    model = '
        eta ~ m_nom;
        eta ~~ eta@1;
        eta -> xc1@lo1 xc2:xc4;
        xc3 ~~ xc4;
        y_bin ~ eta m_nom;',
    seed = 90291,
    burn = 20000,
    iter = 20000
)
output(model7a)
```

MODEL 7B: Multicategorical Outcome Model

Model Specification

- Multicategorical outcome variable (y_nom)
- Multicategorical exogenous predictor (m_nom)
- Ordinal factor indicators (xc1:xc4)
- Correlated indicator residuals (xc3, xc4)

In this example, we extend the binary outcome model to a multicategorical outcome. By specifying `y_nom` in the `nominal` input, Blimp will treat this variable as a multicategorical outcome and use a multinomial probit model. The lowest numeric code is the reference group. For a variable with K categories, K-1 outcome equations are estimated.

```
# Fit model with multicategorical outcome
model7b <- rblimp(
    data = data1,
    ordinal = 'xc1:xc4',
    nominal = 'm_nom y_nom',
    latent = 'eta',
    model = '
        eta ~ m_nom;
        eta ~~ eta@1;
```

```
        eta -> xc1@lo1 xc2:xc4;
        xc3 ~~ xc4;
        y_nom ~ eta m_nom;',
    seed = 90291,
    burn = 20000,
    iter = 20000
)
output(model7b)
```

MODEL 8: Count Outcome Model

Model Specification

- Count outcome variable (y_cnt)
- Multicategorical exogenous predictor (m_nom)
- Ordinal factor indicators (xc1:xc4)
- Correlated indicator residuals (xc3, xc4)

In this example, we illustrate how to specify a count outcome variable. By specifying `y_cnt` in the `count` input, Blimp will treat this variable as a count outcome and use a negative binomial regression model.

```
# Fit model with count outcome using Poisson distribution
model8 <- rblimp(
    data = data1,
    ordinal = 'xc1:xc4',
    nominal = 'm_nom',
    count = 'y_cnt',
    latent = 'eta',
    model = '
        eta ~ m_nom;
        eta ~~ eta@1;
        eta -> xc1@lo1 xc2:xc4;
        xc3 ~~ xc4;
        y_cnt ~ eta m_nom;',
    seed = 90291,
    burn = 25000,
    iter = 25000
)
output(model8)
```

Models 9-12: Advanced Latent Variable Models

Models 9-12 demonstrate advanced latent variable modeling techniques. These models introduce more complex concepts including:

- Nonnormal factor indicators with transformations
- Heterogeneous factor variance across groups
- Latent variable interactions
- Curvilinear effects and moderation

These models extend the basic latent variable framework to handle more complex data structures and research questions.

MODEL 9: Nonnormal Factor Indicators Model

Model Specification

- Normal outcome variable
- Correlated latent and binary manifest predictors
- Nonnormal factor indicators with Yeo-Johnson transformation
- Correlated indicator residuals (xs3, xs4)

In this example, we illustrate how to handle nonnormal factor indicators. When factor indicators are skewed, Blimp can apply a Yeo-Johnson transformation (`yjt()`) to normalize the distribution. This transformation is estimated as part of the model and can characterize positive or negative skew.

```
# Fit model with nonnormal factor indicators using transformations
model9 <- rblimp(
    data = data1,
    ordinal = 'm_bin',
    latent = 'eta',
    model = '
        eta ~~ m_bin;
        eta ~~ eta@1;
        eta -> yjt(xs1)@lo1 yjt(xs2:xs4);
        xs3 ~~ xs4;
        y ~ eta m_bin;',
    seed = 90291,
    burn = 10000,
    iter = 10000
)
output(model9)
```

MODEL 10: Heterogeneous Factor Variance Model

Model Specification

- Normal outcome variable
- Multicategorical exogenous predictor (m_nom)
- Factor variance varies by group
- Ordinal factor indicators (xc1:xc4)
- Correlated indicator residuals (xc3, xc4)

In this example, we illustrate how to model heterogeneous factor variance across groups. Instead of assuming the latent variable has the same variance across all groups, we allow the variance to differ by the nominal predictor `m_nom`. This is specified by regressing the variance of `eta` on `m_nom`. The variance of the factor is fixed to 1 (i.e., log of zero = 1) for identification, and the effects of `m_nom` represent the difference in variance between the reference group.

```
# Fit model with factor variance varying by group
model10 <- rblimp(
    data = data1,
    ordinal = 'xc1:xc4',
    nominal = 'm_nom',
    latent = 'eta',
    model = '
        eta ~ m_nom;
        var(eta) ~ intercept@0 m_nom;
        eta -> xc1@x1lo xc2:xc4;
        xc3 ~~ xc4;
```

```
        y ~ eta m_nom;',
    seed = 90291,
    burn = 20000,
    iter = 20000
)
output(model10)
```

MODEL 11: Interaction Model

Model Specification

- Normal outcome variable
- Multicategorical exogenous predictor (m_nom)
- Multicategorical by latent interaction (eta*m_nom)
- Ordinal factor indicators (xc1:xc4)
- Correlated indicator residuals (xc3, xc4)
- Simple slopes analysis included

In this example, we illustrate how to specify an interaction between a latent variable and a nominal predictor. The interaction is specified using `eta*m_nom` in the model syntax. This will create K-1 interaction terms for a nominal variable with K categories. The `simple` parameter requests simple slopes analysis, which tests the effect of `eta` at each level of `m_nom`.

```
# Fit model with latent by nominal interaction
model11 <- rblimp(
    data = data1,
    ordinal = 'xc1:xc4',
    nominal = 'm_nom',
    latent = 'eta',
    model = '
        eta ~ m_nom;
        eta ~~ eta@1;
        eta -> xc1@lo1 xc2:xc4;
        xc3 ~~ xc4;
        y ~ eta m_nom eta*m_nom;',
    simple = 'eta | m_nom',
    seed = 90291,
    burn = 20000,
    iter = 20000
)
output(model11)
```

`rblimp` can also produce plots of the conditional effects. Using the `simple_plot()` function below, we obtain the regression of `y` on `eta` given different values of the nominal moderator. For continuous interactions, the `jn_plot()` function can be used to obtain the Johnson-Neyman plots.

```
# Plot of simple effects
simple_plot(y ~ eta | m_nom.1 + m_nom.2, model11)
```

MODEL 12: Curvilinear Moderation Model

Model Specification

- Normal outcome variable
- Multicategorical exogenous predictor (m_nom)
- Multicategorical by latent interaction

- Curvilinear effect of latent variable moderated by grouping variable
- Ordinal factor indicators (xc1:xc4)
- Correlated indicator residuals (xc3, xc4)

In this example, we extend the interaction model to include curvilinear effects. We specify both linear (`eta`) and quadratic (`eta^2`) effects of the latent variable, and allow both to be moderated by the nominal predictor `m_nom`. This allows for complex nonlinear relationships that differ across groups. The syntax `(eta^2)*m_nom` creates the quadratic interaction terms.

```
# Fit model with curvilinear latent effect moderated by group
model12 <- rblimp(
    data = data1,
    ordinal = 'xc1:xc4',
    nominal = 'm_nom',
    latent = 'eta',
    model = '
        eta ~ m_nom;
        eta ~~ eta@1;
        eta -> xc1@x1lo xc2:xc4;
        xc3 ~~ xc4;
        y ~ eta eta^2 m_nom eta*m_nom (eta^2)*m_nom;',
    seed = 90291,
    burn = 20000,
    iter = 20000
)
output(model12)
```

Models 13-15: Growth and Sum Score Models

Models 13-15 introduce specialized modeling techniques including:

- Linear growth models with autoregressive residuals
- Sum score predictors as alternatives to latent variables
- Interactions with sum scores

These models provide alternative approaches to modeling change over time and simplifying measurement models when appropriate.

MODEL 13: Linear Growth Model

Model Specification

- Linear growth with four normal repeated measurements (x1-x4)
- Binary exogenous predictor (m_bin)
- AR(1) structure on residuals
- Normal distal outcome (y)
- Latent variables: alpha (intercept), eta (slope)

In this example, we illustrate how to specify a linear growth model with autoregressive residuals. The latent variables `alpha` and `eta` represent the intercept and slope of the growth trajectory. The repeated measures (x1-x4) load on both latent variables with fixed loadings (0, 1, 2, 3) for linear growth. The AR(1) structure is specified by regressing each time point on the residual from the previous time point, with a common autoregressive parameter `rho`.

```
# Fit linear growth model with autoregressive residuals
model13 <- rblimp(
    data = data2,
```

```
    latent = 'alpha eta',
    model = '
        intercept m_bin -> alpha eta;
        alpha ~~ eta;
        x1 ~ intercept@alpha eta@0;
        x2 ~ intercept@alpha eta@1 (x1 - alpha - 0*eta)@rho;
        x3 ~ intercept@alpha eta@2 (x2 - alpha - 1*eta)@rho;
        x4 ~ intercept@alpha eta@3 (x3 - alpha - 2*eta)@rho;
        y ~ m_bin alpha eta',
    seed = 90291,
    burn = 10000,
    iter = 10000
)
output(model13)
```

MODEL 14: Sum Score Predictor Model

Model Specification

- Normal outcome variable
- Sum score predictor (xc1:xc4)
- Binary manifest predictor (m_bin)
- Ordinal indicators (xc1:xc4)

In this example, we illustrate how to use sum scores as predictors instead of latent variables. Sum scores provide a simpler alternative to latent variable models when measurement error is not a primary concern. The syntax `xc1:+:xc4` is a shortcut for specifying `xc1 + xc2 + xc3 + xc4`. Blimp will automatically handle missing data in the sum score calculation, using available data for each case.

```
# Fit model using sum score as predictor
model14 <- rblimp(
    data = data1,
    ordinal = 'm_bin xc1:xc4',
    model = '
        y ~ xc1:+:xc4 m_bin;',
    seed = 90291,
    burn = 20000,
    iter = 20000
)
output(model14)
```

MODEL 15: Sum Score Interaction Model

Model Specification

- Normal outcome variable
- Sum score predictor (xc1:xc4)
- Binary manifest predictor (m_bin)
- Sum score by binary interaction
- Ordinal indicators (xc1:xc4)

In this example, we extend the sum score model to include an interaction between the sum score and a nominal predictor. The syntax `(xc1:+:xc4)*m_bin` creates the product between the two. This allows the effect of the sum score to differ across levels of the binary predictor.

```
# Fit model with sum score by binary interaction
model15 <- rblimp(
    data = data1,
    ordinal = 'xc1:xc4',
    nominal = 'm_bin',
    model = '
        y ~ xc1:+:xc4 m_bin (xc1:+:xc4)*m_bin;',
    seed = 90291,
    burn = 20000,
    iter = 20000
)
output(model15)
```

Models 16-19: Multilevel Models

Models 16-19 introduce multilevel (hierarchical) modeling with:

- Two-level random effects models
- Explicit and implicit random effects specifications
- Cross-level interactions
- Group mean centering
- Heterogeneous within-cluster variation

These models extend the basic framework to handle nested data structures such as students within schools or repeated measures within individuals.

MODEL 16: Two-Level Model with Explicit Latent Variables

Model Specification

- Two-level random slope model
- Normal outcome variable
- Binary exogenous predictor (m_bin)
- Explicit latent variables at level 2 (alpha, eta)
- Latent group means (x.mean)
- Cross-level interaction
- Binary by latent group mean interaction

In this example, we illustrate how to specify a two-level random effects model with explicit latent variables. The `clusterid` parameter identifies the level-2 clustering variable. The `latent = 'l2id = alpha eta'` syntax creates level-2 latent variables representing the random intercept (`alpha`) and random slope (`eta`). The `center = 'groupmean = x'` automatically group mean centers predictor variables. By appending `.mean` to a variable, it automatically uses latent cluster means for that variable. This explicit specification mimics the two-level notation often used in multilevel models.

```
# Fit two-level model with explicit random intercept and slope
model16 <- rblimp(
    data = data3,
    nominal = 'm_bin',
    clusterid = 'l2id',
    latent = 'l2id = alpha eta',
    center = 'groupmean = x',
    model = '
        alpha ~ intercept x.mean m_bin x.mean*m_bin;
        eta ~ intercept m_bin;
```

```
        alpha ~~ eta;
        y ~ intercept@alpha x@eta;',
    seed = 90291,
    burn = 10000,
    iter = 10000
)
output(model16)
```

MODEL 17: Two-Level Random Slope Model

Model Specification

- Two-level random slope model
- Normal outcome variable
- Binary exogenous predictor (m_bin)
- Latent group means (x.mean)
- Cross-level interaction
- Binary by latent group mean interaction

In this example, we illustrate an alternative implicit specification for two-level models. Instead of explicitly defining latent variables for random effects, we use the `|` syntax to specify random effects. The syntax `y ~ ... | x` indicates that `x` has a random slope. Blimp will automatically create random intercept and slope latent variables. This provides a more compact syntax that mimics mixed model notation.

```
# Fit two-level random slope model with cross-level interaction
model17 <- rblimp(
    data = data3,
    nominal = 'm_bin',
    clusterid = 'l2id',
    center = 'groupmean = x',
    model = '
        y ~ x x*m_bin x.mean m_bin x.mean*m_bin | x;',
    seed = 90291,
    burn = 10000,
    iter = 10000
)
output(model17)
```

MODEL 18: Heterogeneous Within-Cluster Variation Model

Model Specification

- Two-level random slope model
- Normal outcome variable
- Binary exogenous predictor (m_bin)
- Latent group means (x.mean)
- Cross-level interaction
- Binary by latent group mean interaction
- Heterogeneous within-cluster variation (HEV option)

In this example, we extend Model 17 to allow for heterogeneous within-cluster variation. By default, multilevel models assume constant within-cluster variance. The `options = 'hev'` parameter relaxes this assumption, allowing the level-1 residual variance to vary across level-2 units. This can improve correct for violations to the homogeneity of variance assumption.

```
# Fit model with heterogeneous within-cluster variance
model18 <- rblimp(
    data = data3,
    nominal = 'm_bin',
    clusterid = 'l2id',
    center = 'groupmean = x',
    model = '
        y ~ x x*m_bin x.mean m_bin x.mean*m_bin | x;',
    seed = 90291,
    burn = 10000,
    iter = 10000,
    options = 'hev'
)
output(model18)
```

MODEL 19: HEV with Predictors at Both Levels

Model Specification

- Two-level random slope model
- Normal outcome variable
- Binary exogenous predictor (m_bin)
- Latent group means (x.mean)
- Cross-level interaction
- Binary by latent group mean interaction
- Heterogeneous within-cluster variation with predictors at level-1 and level-2
- Three latent variables: alpha (intercept), eta (slope), omega (residual variance)

In this example, we extend the HEV model to include predictors of the within-cluster variance. A third latent variable `omega` represents the log of the level-1 residual variance, which can be predicted by both level-1 and level-2 variables. The syntax `var(y) ~ intercept@omega x` specifies that the log variance of `y` is a linear combination of `omega` (intercept) and the level-1 predictor `x`. This allows for a more generalized location-scale modeling framework at both levels.

```
# Fit HEV model with variance predictors
model19 <- rblimp(
    data = data3,
    nominal = 'm_bin',
    clusterid = 'l2id',
    latent = 'l2id = alpha eta omega',
    center = 'groupmean = x',
    model = '
        alpha ~ intercept x.mean m_bin x.mean*m_bin;
        eta ~ intercept m_bin;
        omega ~ intercept x.mean m_bin;
        alpha eta omega ~~ alpha eta omega;
        y ~ intercept@alpha x@eta;
        var(y) ~ intercept@omega x;',
    seed = 90291,
    burn = 10000,
    iter = 10000
)
output(model19)
```

Models 20-22: Advanced Longitudinal Models

Models 20-22 introduce specialized longitudinal modeling techniques including:

- Two-part models for semi-continuous outcomes
- Random intercept cross-lagged panel models (RI-CLPM)
- Dynamic structural equation models for intensive longitudinal data

These models provide advanced techniques for handling complex data structures and temporal dependencies.

MODEL 20: Two-Part Outcome Model

Model Specification

- Two-part outcome variable (binary indicator u, continuous yr)
- Multicategorical exogenous predictor (m_nom)
- Ordinal factor indicators (xc1:xc4)
- Correlated indicator residuals (xc3, xc4)
- Data transformation to create binary and continuous parts

In this example, we illustrate how to specify a two-part model for semi-continuous outcomes (e.g., healthcare costs, alcohol consumption). The `transform` parameter creates two variables: `u` (binary indicator of whether the outcome is zero) and `yr` (the continuous positive part, with zeros set to missing). The model then estimates separate equations for the binary and continuous parts. The syntax `missing(u == 0, y_2pt)` sets `yr` to missing when `u` equals 0.

```
# Fit two-part model for semi-continuous outcome
model20 <- rblimp(
    data = data1,
    ordinal = 'u xc1:xc4',
    nominal = 'm_nom',
    latent = 'eta',
    transform = '
        u = ifelse(y_2pt == min(y_2pt), 0, 1);
        yr = missing(u == 0, y_2pt);',
    model = '
        eta ~ m_nom;
        eta -> xc1:xc4;
        xc3 ~~ xc4;
        yr ~ eta m_nom;
        u ~ eta m_nom;',
    seed = 90291,
    burn = 20000,
    iter = 20000
)
output(model20)
```

MODEL 21: Random Intercept Cross-Lagged Panel Model

Model Specification

- Random intercept cross-lagged panel model (RI-CLPM)
- Three waves of bivariate data (x1-x3, y1-y3)
- Latent variables: alpha (random intercept for x), eta (random intercept for y)
- Cross-lagged effects between x and y
- Correlated residuals within time points

In this example, we illustrate how to specify a random intercept cross-lagged panel model (RI-CLPM). This model separates between-person and within-person effects in longitudinal data. The latent variables `alpha` and `eta` represent stable between-person differences (random intercepts) for x and y. The cross-lagged effects are estimated using within-person deviations from these intercepts. The syntax `(x1 - (mux1 + alpha))` creates the within-person deviation for x at time 1.

Model 21a: RI-CLPM with direct specification

```
model21a <- rblimp(
    data = data4,
    latent = ' alpha eta',
    model = '
        alpha ~~ eta;
        x1 ~ intercept@mux1 alpha@1;
        x2 ~ intercept@mux2 alpha@1 (x1 - (mux1 + alpha)) (y1 - (muy1 + eta));
        x3 ~ intercept@mux3 alpha@1 (x2 - (mux2 + alpha)) (y2 - (muy2 + eta));
        y1 ~ intercept@muy1 eta@1;
        y2 ~ intercept@muy2 eta@1 (y1 - (muy1 + eta)) (x1 - (mux1 + alpha));
        y3 ~ intercept@muy3 eta@1 (y2 - (muy2 + eta)) (x2 - (mux2 + alpha));
        x1 ~~ y1;
        x2 ~~ y2;
        x3 ~~ y3;',
    seed = 90291,
    burn = 10000,
    iter = 10000
)
output(model21a)
```

An alternative specification creates explicit residual variables by defining a variable within the model syntax. This approach is identical to the previous approach, but offers a more clear picture on what is happening.

Model 21b: Alternative RI-CLPM specification using residuals

```
model21b <- rblimp(
    data = data4,
    latent = ' alpha eta',
    model = '
        # Define Residuals
        x1res = x1 - (mux1 + alpha);
        x2res = x2 - (mux2 + alpha);
        y1res = y1 - (muy1 + eta);
        y2res = y2 - (muy2 + eta);

        # Specify Model
        alpha ~~ eta;
        x1 ~ 1@mux1 alpha@1;
        x2 ~ 1@mux2 alpha@1 x1res y1res;
        x3 ~ 1@mux3 alpha@1 x2res y2res;
        y1 ~ 1@muy1 eta@1;
        y2 ~ 1@muy2 eta@1 y1res x1res;
        y3 ~ 1@muy3 eta@1 y2res x2res;
        x1 ~~ y1;
        x2 ~~ y2;
        x3 ~~ y3;',
```

```
    seed = 90291,
    burn = 10000,
    iter = 10000
)
output(model21b)
```

MODEL 22: Dynamic Structural Equation Model

Model Specification

- Dynamic SEM with intensive longitudinal data
- Level 2 cluster ID and time ID specified
- Latent variables at level 2: alpha, eta
- Lagged predictors (x.lag, y.lag)
- Cross-lagged effects between x and y
- Correlated residuals

In this example, we illustrate how to specify a dynamic structural equation model (DSEM) for intensive longitudinal data (e.g., ecological momentary assessment, daily diary studies). The key feature is the `timeid` specification in the `clusterid` parameter, which tells Blimp to automatically create lagged variables (`.lag` suffix). The model includes both between-person random intercepts (`alpha`, `eta`) and within-person cross-lagged effects.

```
# Fit dynamic SEM with lagged predictors
model22 <- rblimp(
    data = data5,
    clusterid = 'l2id; timeid: time;',
    latent = 'l2id = alpha eta',
    model = '
        x ~ intercept@alpha (x.lag - alpha) (y.lag - eta);
        y ~ intercept@eta (y.lag - eta) (x.lag - alpha);
        x ~~ y;
        intercept -> alpha eta;
        alpha ~~ eta;',
    seed = 90291,
    burn = 10000,
    iter = 10000
)
output(model22)
```