Mplus Tutorial for New and Advanced Users

AS F!

Ryne Estabrook

Department of Psychology
University of Virginia

Design and Data Analysis
February 19, 2009
Why should we talk about Mplus?

- If you’re not familiar, it’s a very flexible program for structural models and a variety of other GLM techniques.
- If you are familiar, it has a lot of capabilities that many people aren’t familiar with.
- If you’re a developer for an alternative program, then understanding popular SEM programming can help shape new techniques and show what people use and expect.
- No one should use only one program. They all have strengths and weaknesses, and knowing a few can help you as a user choose the correct one for the problem at hand.

Today’s talk:

- Should be flexible, so we can skip around or not finish certain sections.
- Includes lots of code, so you can take it and use it as a primer if you like.
Mplus

Topic of the Day

▶ Mplus is a program for structural equation modeling utilizing code-based path-centric specification.
▶ It consists of a base package and “add-ons” for mixture and multilevel modeling.
▶ It uses a Windows GUI and a DOS backend running either 32 or 64 bit processing.
  ▶ 64-bit Mplus is not subject to the Windows memory limit, so there’s no restriction on model size.
  ▶ It can also be called from DOS, and run with batch files.
▶ It takes input files with the extension .inp and returns .out files, although additional output options are available.
▶ You can get more information, the manual and the message boards at www.statmodel.com.
Mplus

Pros

▶ The path-centric specification is relatively intuitive and easy to teach.
▶ Being code based, extensions to larger models are easier to build than graphical path-centric specification (Ctrl+C).
▶ It has some decidedly non-traditional features, and appears committed to constant development.
▶ User support is good (Bengt & Linda Muthén personally respond to most, if not all, message board threads).
Cost.
Matrix specification not supported (to my knowledge).
Black box.
Code can get unwieldy at times.
No data management beyond Monte Carlo capabilities.
Does rely on defaults.
This is why you learn multiple programs.
Basics of Mplus

- Mplus commands are written in scripts with sections known as commands.
- These commands include:
  - TITLE:
  - DATA:
  - VARIABLE:
  - ANALYSIS:
  - MODEL:
  - OUTPUT:
- Other commands that we’ll discuss, time permitting:
  - DEFINE:
  - SAVEDATA:
  - PLOT:
  - MONTECARLO:
- The comment character is !, which I’ll try to use extensively.
The **TITLE**: Command

- This makes titles.

**TITLE**: This is the title of my project;

- Mplus has in the past (and may currently) impose a width limit of 80 characters on input files.
The **DATA**: Command

```plaintext
DATA: FILE=mydata.dat;
    TYPE=INDIVIDUAL; !DEFAULT
    !TYPE=COVARIANCE;
```

- The general usage of Mplus uses tab or space delimited individual data, with a 500 variable limit.
- It can also accept covariance or correlation matrices, fixed format data, monte carlo and imputation components, as well as multiple data set specification.
- Data should be in the same folder as the input file.
The **VARIABLE**: Command

**VARIABLE:**

```
NAMES = ID x1-x3 y1-y3 z1-z3;
USEVARIABLES = x1-x3 y1-y3;
```

- The required statements are **NAMES**, though **USEVAR** may as well be.
  - Variables specified in the **USEVAR** statement are those specifically used in the covariance calculation, so ID & related variables aren’t included.
  - Every column in the data must be named.
  - The equal sign and **ARE** can be used interchangeably, but the equal sign can’t be confused for a variable name.

- Other statements include:
  - **CATEGORICAL**, **NOMINAL & CENSORED**
  - **MISSING**, which specifies missing data symbols
  - **IDVARIABLE**, **GROUPING**, **CLUSTER**, etc.
  - And many more analysis-specific statements.
The **VARIABLE**: Command

A Note About the Dash

```
VARIABLE:
    NAMES = ID x1-x3 y1-y3 z1-z3;
    USEVARIABLES = x1-x3 y1-y3;
```

- The code above uses dashes to gloss over repetition.
  - `x1-x3` refers to x1, x2 and x3.
  - No leading zeros.

- Outside of (after) the **VARIABLE**: command, these will refer to the variable array (list) as defined by **USEVARIABLES**
  - Referring to `x1-y3` would refer to x1, x2, x3, y1, y2, & y3.
  - Had I listed the variables in the **USEVARIABLES** statement as `x1 y3 y2 x3 x2 y1`, stating `x1-x3` would reference x1, y3, y2 and x3.
The **ANALYSIS**: Command

- The **ANALYSIS**: command is used for specifying specific analysis types and changing estimation procedures.
- Fun topics include:
  - The **TYPE** statement, which specifies analysis types (BASIC, EFA, MIXTURE, TWOLEVEL).
  - The **ESTIMATOR** statement (flavors of ML and least squares).
  - The **INTEGRATION** statement, for affecting integration techniques for ML (STANDARD, GAUSSHERMITE, MONTECARLO).
The **OUTPUT**: Command

Hey, you skipped **MODEL**: 

- The **OUTPUT**: command is to control what is output.
- When it runs, Mplus creates an output file using the extension `.out`.
- It typically includes:
  - Estimation and model summaries.
  - Covariance coverage.
  - Model fit statistics.
  - Parameter estimates and standard errors.
- A number of things can be included or suppressed, spanning statistical and technical output.
The **MODEL**: Command

- This is where Mplus becomes a language.
- Models are built in terms of relationships between variables.
- The general structure is as follows:

  ```
  Variable RELATIONSHIP Variable (label);
  ```

- The label is always optional, and is of more use for constraints than for actual labeling.
Variances are pretty easy. You just name the variable.

\[ X_1; \]
Covariances

- Covariances are specified using `WITH`, as in “this covaries with that.”
Regression

- Regressions are specified using `ON`, as in “this is regressed on that.”
- The first variable is the dependent variable, which is regressed on the independent variable.

```
Y1 ON X1;
```
Means

► The means of variables are specified by placing that variable’s name inside square brackets.
► Path-tracing rules apply.
► Manifest variable means are often estimated by default.

\[
[Y_1];
\]
We now have all of the tools we need to estimate the following model:
Example

▶ Here’s the **MODEL** statement for that model:

```plaintext
MODEL:
  !Regressions
  Y1 ON X1-X2;
  !Covariances
  X1 WITH X2;
  !Variances
  X1; X2; Y1;
  !Means;
  [X1 X2 Y1];
```
Example

Here’s the full code for that model:

```
TITLE: DADA’s first model;
DATA: file=mydata.txt;
VARIABLE: NAMES = ID sex x1-x3 y1-y3 z1-z3;
    USEVAR= x1 x2 y1;
MODEL:
  !Regressions
  Y1 ON X1-X2;
  !Covariances
  X1 WITH X2;
  !Variances
  X1; X2; Y1;
  !Means;
  [X1 X2 Y1];
OUTPUT: SAMPSTAT STANDARDIZED;
```
SUMMARY OF ANALYSIS

Number of groups 1
Number of observations 100

Number of dependent variables 1
Number of independent variables 2
Number of continuous latent variables 0

Observed dependent variables

Continuous
Y1

Observed independent variables
X1 X2

Estimator ML
Information matrix OBSERVED
Maximum number of iterations 1000
Convergence criterion 0.500D-04
Maximum number of steepest descent iterations 20

Input data file(s)
mydata.txt

Input data format FREE
## Sample Statistics

### Means

<table>
<thead>
<tr>
<th></th>
<th>Y1</th>
<th>X1</th>
<th>X2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-0.148</td>
<td>0.673</td>
<td>0.638</td>
</tr>
</tbody>
</table>

### Covariances

<table>
<thead>
<tr>
<th></th>
<th>Y1</th>
<th>X1</th>
<th>X2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y1</td>
<td>0.894</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X1</td>
<td>0.689</td>
<td>1.672</td>
<td></td>
</tr>
<tr>
<td>X2</td>
<td>0.748</td>
<td>1.430</td>
<td>2.024</td>
</tr>
</tbody>
</table>

### Correlations

<table>
<thead>
<tr>
<th></th>
<th>Y1</th>
<th>X1</th>
<th>X2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y1</td>
<td>1.000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X1</td>
<td>0.564</td>
<td>1.000</td>
<td></td>
</tr>
<tr>
<td>X2</td>
<td>0.556</td>
<td>0.778</td>
<td>1.000</td>
</tr>
</tbody>
</table>

This is what was requested with the `SAMPSTAT` statement from the `OUTPUT` command.
Output from the Example, Part III

THE MODEL ESTIMATION TERMINATED NORMALLY

TESTS OF MODEL FIT

Chi-Square Test of Model Fit

<table>
<thead>
<tr>
<th>Value</th>
<th>0.000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Degrees of Freedom</td>
<td>0</td>
</tr>
<tr>
<td>P-Value</td>
<td>0.000</td>
</tr>
</tbody>
</table>

Chi-Square Test of Model Fit for the Baseline Model

<table>
<thead>
<tr>
<th>Value</th>
<th>43.529</th>
</tr>
</thead>
<tbody>
<tr>
<td>Degrees of Freedom</td>
<td>2</td>
</tr>
<tr>
<td>P-Value</td>
<td>0.000</td>
</tr>
</tbody>
</table>

CFI/TLI

<table>
<thead>
<tr>
<th>CFI</th>
<th>1.000</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLI</td>
<td>1.000</td>
</tr>
</tbody>
</table>

Loglikelihood

<table>
<thead>
<tr>
<th>H0 Value</th>
<th>-412.847</th>
</tr>
</thead>
<tbody>
<tr>
<td>H1 Value</td>
<td>-412.847</td>
</tr>
</tbody>
</table>
Output from the Example, Part IV

Information Criteria

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Free Parameters</td>
<td>9</td>
</tr>
<tr>
<td>Akaike (AIC)</td>
<td>843.695</td>
</tr>
<tr>
<td>Bayesian (BIC)</td>
<td>867.141</td>
</tr>
<tr>
<td>Sample-Size Adjusted BIC</td>
<td>838.717</td>
</tr>
<tr>
<td>(n* = (n + 2) / 24)</td>
<td></td>
</tr>
</tbody>
</table>

RMSEA (Root Mean Square Error Of Approximation)

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Estimate</td>
<td>0.000</td>
</tr>
<tr>
<td>90 Percent C.I.</td>
<td>0.000</td>
</tr>
<tr>
<td>Probability RMSEA &lt;= .05</td>
<td>0.000</td>
</tr>
</tbody>
</table>

SRMR (Standardized Root Mean Square Residual)

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>0.000</td>
</tr>
</tbody>
</table>

➤ That’s the standard set of model fit statistics for Mplus.
### Output from the Example, Part V

#### MODEL RESULTS

<table>
<thead>
<tr>
<th></th>
<th>Estimate</th>
<th>S.E.</th>
<th>Est./S.E.</th>
<th>Two-Tailed P-Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Y1 ON</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Y1 X1</td>
<td>0.242</td>
<td>0.094</td>
<td>2.588</td>
<td>0.010</td>
</tr>
<tr>
<td>Y1 X2</td>
<td>0.199</td>
<td>0.085</td>
<td>2.336</td>
<td>0.019</td>
</tr>
<tr>
<td><strong>X1 WITH</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X1 X2</td>
<td>1.430</td>
<td>0.233</td>
<td>6.139</td>
<td>0.000</td>
</tr>
<tr>
<td><strong>Means</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X1</td>
<td>0.673</td>
<td>0.129</td>
<td>5.204</td>
<td>0.000</td>
</tr>
<tr>
<td>X2</td>
<td>0.638</td>
<td>0.142</td>
<td>4.488</td>
<td>0.000</td>
</tr>
<tr>
<td><strong>Intercepts</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Y1</td>
<td>-0.438</td>
<td>0.086</td>
<td>-5.097</td>
<td>0.000</td>
</tr>
<tr>
<td><strong>Variances</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X1</td>
<td>1.672</td>
<td>0.236</td>
<td>7.071</td>
<td>0.000</td>
</tr>
<tr>
<td>X2</td>
<td>2.023</td>
<td>0.286</td>
<td>7.071</td>
<td>0.000</td>
</tr>
<tr>
<td><strong>Residual Variances</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Y1</td>
<td>0.578</td>
<td>0.082</td>
<td>7.071</td>
<td>0.000</td>
</tr>
</tbody>
</table>

▶ The **STANDARDIZE** statement from the **OUTPUT** command requested a standardized version of this.
Output from the Example, Part VI

R-SQUARE

<table>
<thead>
<tr>
<th>Observed Variable</th>
<th>Estimate</th>
<th>S.E.</th>
<th>Est./S.E.</th>
<th>Two-Tailed P-Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y1</td>
<td>0.353</td>
<td>0.077</td>
<td>4.590</td>
<td>0.000</td>
</tr>
</tbody>
</table>

QUALITY OF NUMERICAL RESULTS

| Condition Number for the Information Matrix | 0.813E-02 |

(ratio of smallest to largest eigenvalue)

Beginning Time: 15:59:25
Ending Time: 15:59:25
Elapsed Time: 00:00:00

▶ And that’s (finally) the end of the code.
Mplus Output for Basic Modeling

- Mplus spits out a lot of output.
- All of this output is essentially in a text file.
  - These sections will always occur in the same order, so these text files can be stripped for their parameters.
  - However, warnings and other runoff can change line numbers, so codes to read .out files should probably be done on a case-by-case basis.
- Any questions about manifest variable models?
Incorporating Latent Variables

- Incorporating latent variables is easy; just refer to them.
- The language for defining latent variables is \( \text{BY} \), as in “\( F_1 \) defined by \( x_1-x_3 \).”
- This links latent variables to their manifest (or latent) definition variables by way of regressions.

\[
\begin{align*}
F & \quad \sigma^2_F \\
X_1 & \quad \sigma^2_1 \\
X_2 & \quad \sigma^2_2 \\
X_3 & \quad \sigma^2_3 \\
X_4 & \quad \sigma^2_4 \\
X_5 & \quad \sigma^2_5 \\
\end{align*}
\]

\[
f_1 \ \text{BY} \ x_1-x_3;
\]
Scaling Latent Variables

- Latent variables require a scale.
- Mplus default will constrain the first regression implied by a \texttt{BY} statement at 1, and allow the factor variance and all other regressions will be freely estimated.
- This is the specification of the factor model on which the default is based:

  \begin{verbatim}
  f1 BY x1-x3;  !Regressions that define factor
  f1;           !Factor Variance
  \end{verbatim}

- Standardized factor loadings can be found by including the \texttt{STANDARDIZE} statement in the \texttt{OUTPUT} command.
Making simple constraints

▶ You may wish to define latent variables differently or otherwise make simple constraints.

▶ We can constrain a particular relationship to be constant.
  ▶ Most commonly, these constants are zero (covariances & regressions) or unity (variances).
  ▶ They’ll also play a role in growth modeling and dynamical systems.

▶ We can force parameters to be freely estimated.
  ▶ Useful when overriding a default.
  ▶ This is how Mplus handles starting value specification.

▶ We can constrain some number of free parameters (relationships) to be equal.

▶ Remember, equality constraints differ in meaning across parameterizations.
Making simple constraints

- Constraints to a constant are made with an @:
  \[ f1@1; \] !Factor Variance Constrained to 1
  \[ x1 \text{ WITH } x2@0; \] !Covariance Constrained to 0

- Free estimation & starting values use a *:
  \[ [f1*]; \] !Factor Mean Freely Estimated
  \[ f1 \text{ BY } x1*1; \] !Regression Freely Estimated, Starting Value=1

- Equality constraints use labels in parentheses:
  \[ x1 \text{ (e)}; \] \[ x2 \text{ (e)}; \] !Variances of x1 and x2 Equal
  \[ x1 \text{ WITH } y1 \text{ (cov)}; \]
  \[ x2 \text{ WITH } y2 \text{ (cov)}; \] !Covariances Equal
Latent and Constraint Example

- We can go through a few model statements for examples involving latent variables & constraints.
- First, a common factor model, using defaults:
  - Factor defined by constraining first referenced regression (x1 in `BY` statement) to 1.
  - Manifest variables have means, latent variable doesn’t.

```plaintext
MODEL:
!Regressions
f1 BY x1-x3;
!Manifest Variances
x1-x3;
!Latent Variances
f1;
```
Let’s refit that same model, blowing off the defaults and defining the factor as having unit variance:

```
MODEL:
!Regressions
f1 BY x1-x3*; !The * is applied to all three loadings
!Manifest Variances
x1-x3;
!Latent Variance Constrained to 1
f1@1;
!Means
[x1 x2 x3]; [f1@0];
```
Latent Growth Curves

Let’s fit a linear growth curve model, considering x1-x3 as three timepoints:

```plaintext
MODEL:
!Regressions
level BY x1-x3@1; !The @1 is distributive
slope BY x1@0; !Slope loading at time 1
slope BY x2@0.5; !Slope loading at time 2
slope BY x3@1; !Slope loading at time 3
!Manifest Variances
X1-X3 (e);
!Latent Variances & Covariances
level; slope;
level WITH slope;
!Means
[x1-x3@0]; [level* slope*];
```
Latent Basis Growth Curves

Let’s fit a latent basis growth curve model. Find the difference:

MODEL:
!Regressions
level BY x1-x3@1;  !The @1 is distributive
slope BY x1@0;  !Slope loading at time 1
slope BY x2*0.5;  !Slope loading at time 2
slope BY x3@1;  !Slope loading at time 3
!Manifest Variances
X1-X3 (e);
!Latent Variances & Covariances
level; slope;
level WITH slope;
!Means
[x1-x3@0]; [level* slope*];
Different Model Constraints

- More complex model constraints can also be done in Mplus.
- The `MODEL CONSTRAINT` statement goes at the end of the `MODEL` command, and specifies relationships between the labeled paths in the model.
  - A large number of functions are available, which we’ll review when we get to transformations later.

```
MODEL:
!Regressions
f2 ON f1 (reg);
!Manifest Variances
f1@1; f2 (var2);
MODEL CONSTRAINT:
var2=1-reg**2;!Variance of Factor 2=1
```
Different Model Constraints

- Variables from the dataset that aren’t in the model may be used in constraints if identified by `CONSTRAINTS` in the `VARIABLE` command.
- New variables can also be created for the purposes of model constraints.

```plaintext
MODEL CONSTRAINT:
NEW (k*1)
var2=var1**k;
var3=var2**k

- Model constraints can be tested using the Wald test with the `MODEL TEST` statement.

x1-x3 (varx); y1-y3 (vary);
MODEL TEST:
varx=vary;
```
Advanced Language Usage

Latent Variable Specification

- There are some things that I need to address regarding use of the language for larger models.
  - Some are necessary, dealing with defaults and program specifics that can crash the program.
  - Some are just nice to know, saving you keystrokes and keeping models intuitive and manageable.
- Latent Variables must be defined before they are used in covariances, as predictors or in definitions of other latent variables.
  - As they’re not defined in the `USEVAR` statement, you have to be a little careful.
  - If you run a second-order factor analysis, you must code the first level first.
Advanced Language Usage
Efficient Coding

- You can put series of variables on both sides of an option, and all possible permutations are employed. For instance:

\[ x_1-x_2 \text{ WITH } y_1-y_3; \]

- is the same as this mess:

\[ x_1 \text{ WITH } y_1; \ x_2 \text{ WITH } y_1; \]
\[ x_1 \text{ WITH } y_2; \ x_2 \text{ WITH } y_2; \]
\[ x_1 \text{ WITH } y_3; \ x_2 \text{ WITH } y_3; \]

- Variables can appear on both sides of a \textit{WITH} statement with no effect but a meaningless warning.
Advanced Language Usage

Pairwise Statements

- You can employ pairwise use of the ON and WITH options by prefixing them with the letter “P.”
- These options must be balanced.
- For example:

  \[
  x1-x3 \text{ PWITH } y1-y3;
  \]

  - is the same as the following:

  \[
  x1 \text{ WITH } y1;
  x2 \text{ WITH } y2;
  x3 \text{ WITH } y3;
  \]

  - This is particularly useful for equality constraints through labeling.
With this small set of commands, you could fit a large majority of the SEM models in use in psychology.

You can leave the `TITLE` and `OUTPUT` lines out in many cases, leaving the `DATA`, `VARIABLE` & `MODEL` commands.

You could do everything but your model specification in three lines of code.

See the manual (available for free) for more information and a host of examples.
Variable-Related Modeling Issues

- Missing Data.
- Data transformation and definitions.
- Categorical Data.
- Multiple Groups.
Missing Data

- Mplus has a few different ways of dealing with missing data, but the most common involves MCAR & MAR assumptions and FIML.
  - The EM algorithm is used here, but some documentation refers to a multiple group approach as well.
- The general way of invoking these assumptions is to specify a character for missing data in the VARIABLE command with the MISSING statement.

```
VARIABLE:  NAMES = ID sex x1-x3 y1-y3 z1-z3;
USEVAR = x1-z3;
MISSING = .;
!MISSING IS .; !Works just as well
```
Missing Data

Notes

- By default, Mplus throws warnings when covariance coverage is below 10%.

- You may specify multiple missingness codes, or different codes for different variables.

MISSING = ALL(-999);!All -999s are missing
MISSING = x1-y3(1 -999) z1-z3(.);!1s and -999s missing for x1-y3; dots missing z1-z3

- Missingness status can be calculated as a binary variable using DATA MISSING:

DATA: FILE=mydata.txt
DATA MISSING: NAMES = x1;
   BINARY = u1;
Transformations
The **DEFINE** command

- One can create new variables with the **DEFINE** command.
  - New variables can be functions of any variable or set of variables in the dataset.
  - To use them in analysis, new variables must be listed in the **USEVAR** list after all preexisting variables.
- These variables function just like any other manifest variable.

```plaintext
VARIABLE:  NAMES = ID sex x1-x3 y1-y3 z1-z3;
          USEVAR = x1-x3 new;
DEFINE:   new=log(z1)*y2;
```
The **DEFINE** command

Supported Functions

- A large number of functions can be used:
  - **Arithmetic**: +, −, *, /, **, % (remainder).
  - **LOG, LOG10, EXP, SQRT, ABS, SIN, COS, TAN, ASIN, ACOS, ATAN**
  - **CUT** can be used to chop a continuous variable into a number of categories:

  ```
  CUT z3 (-1 0 1); !Yields 4 categories for z3
  ```

- You can also define new variables conditionally:
  - **AND, OR, NOT, EQ(==), NE(/=), GE(>=), LE(<=), GT(>), LT(<)**
  - **IF, THEN**

  ```
  IF (x1 GE 0 AND sex NE 1) THEN test=14;
  ```
Categorical Data

- A lot of the original work to come through Mplus has involved its categorical data handling.
- Four statements in the `VARIABLE` command deal with this:
  - `CATEGORICAL` (binary and ordinal variables),
  - `NOMINAL`,
  - `CENSORED`, and
  - `COUNT`
- In general, Mplus deals with categorical data parameters using `#` to designate special properties like thresholds, categories and inflation.
- Categorical data is automatically recoded in ascending integers, starting with zero.

```
u1#1 !Threshold between categories 0 and 1
```
CATEGORICAL Data

- The most common category for categorical data is CATEGORICAL.

VARIABLE: NAMES = ID sex x1-x3 y1-y3 z1-z3;
USEVAR = x1-z3;
CATEGORICAL = z1-z3;

- This specifies some variables (z1-z3) as ordinal (binary if only two categories are found).
  - These variables can be referred to like any other.
  - They can’t have more than 10 unique categories.
  - Invoking this statement will likely change your estimator.
- You have some control over parameterization, particularly for binary items.
  - In the DELTA parameterization, scale factors (loadings) are parameters, residual variances are not (default).
  - The THETA parameterization is the opposite.
NOMINAL Data

- NOMINAL data functions very similarly to CATEGORICAL data.

VARIABLE:  NAMES = ID sex x1-x3 y1-y3 z1-z3;
            USEVAR = x1-z3;
            NOMINAL = z1-z3;

- You can refer to all but the last (reference) category.
  - These seem to be more commonly used for path models as opposed to more advanced and latent variable models.
  - I don’t know of a way to define the reference group.
CENSORED Data

- CENSORED data requires an additional argument, specifying the direction of censoring and whether inflated models will be used.

VARIABLE: NAMES = ID sex x1-x3 y1-y3 z1-z3;
USEVAR = x1-z3;
CENSORED = z1 (a) z2 (b) z3 (ai);

- Above (a) & below (b) refer to censoring at the top and bottom of the range of data, with the suffix “i” indicating inflation.

  - Inflated variable modeling creates an additional binary latent variable for analysis, referenced by variable#1.
  - Model assumptions for censored data should be known and considered in advance.
COUNT Data

- **COUNT data** is useful for specifying variables for Poisson and zero-inflated Poisson models.

```plaintext
VARIABLE: NAMES = ID sex x1-x3 y1-y3 z1-z3;
USEVAR = x1-z3;
COUNT = z1 z2 (i) z3;
```

- Simply listing the variable treats it as a count variable for other analyses.

- The suffix (i) functions similarly to the **Censored** example, creating a binary latent variable that can be referenced and modeled by variable#1.
Multiple Group Modeling

- Mplus supports multiple group analysis for virtually all of its analysis types.
  - Multiple group EFAs aren’t available, but that’s easy enough to work around.
  - Mixture models don’t support multiple groups, but do have a `KNOWNCLASS` option that functions similarly.
- Grouping is specified with the `GROUPING` statement in the `VARIABLE` command.
  - This variable is not a part of the `USEVAR` array.
  - All levels of the variable must be named for later reference.
  - Multiple datasets are also supported.

```plaintext
VARIABLE: NAMES = ID sex x1-x3 y1-y3 z1-z3;
USEVAR = x1-z3;
GROUPING = sex (1=male, 2=female);
```
Multiple Group Modeling

- When a multiple group model is requested, all groups are fit with the model specified in the `MODEL` command.
- One can also specify group specific model commands.
  - The `MODEL` command specifies the general model.
  - Group specific models are specified with `MODEL groupname;`

```
MODEL:
  f1 BY x1-z3;
  x1-z3; f1;
MODEL male;
  f1 BY x1-y3;
  f2 BY z1-z3;
  x1-z3; f1; f2;
```
Multiple Group Modeling Defaults

Mplus likes to guess what you mean

- All groups are fit with the same model by default, but not all parameters are held constant across groups.
- These parameters are equal by default across groups:
  - Factor loadings.
  - Intercepts & thresholds for factor indicators.
  - Think (strong) measurement invariance.
- These parameters freely vary by default across groups:
  - Manifest residual variances.
  - Latent means, variances and covariances.
  - Latent means for “first” group is zero; all others free.
- Equalities can still be done using labels.
- Free parameters in group-specific model statements override defaults.
Multiple Group Modeling Defaults

Two-Group Example Using Sex

MODEL:
  f1  BY  x1-x3;  !Constant
  f2  BY  y1-y3;  !Across Groups
  f3  BY  z1-z3;  !By Default
  [x1-z3@0];
MODEL male;
  f1  f1  f2  PWITH  f2  f3  f3  (1-3);
  [f1*  f2*  f3*];
  f1-f3  (4-6);  x1-z3  (11-19);
MODEL female;
  f1  f1  f2  PWITH  f2  f3  f3  (1-3);
  [f1*  f2*  f3*];
  f1-f3  (4-6);  x1-z3  (11-19);

- This will test for group differences in factor means.
Review and Questions

- Mplus relies on the `VARIABLE` command to alter the types of models it can run.
  - It handles missing data reasonably well.
  - You can define new variables without affecting your data.
  - You can actually handle non-continuous items.
  - Multiple groups is very easy to do, although constraints can be a problem.

- Using these options will change the models you run.
  - Every model has an implied default estimator, though many other options are available.
  - Categorical data tends to employ variants of LS, which change model output, statistics and greatly affect computation times.
Alternative Modeling Topics

- Some models not often thought of from a structural modeling framework can be estimated using Mplus.
- These typically specifying a `TYPE` in the `ANALYSIS` command.
- They co-opt the language of the `MODEL` command whenever possible, and have a few extra components.
  - Exploratory Factor Analysis (`TYPE=EFA`).
  - Multilevel and Mixed Modeling (`TYPE=RANDOM`).
  - Mixture Distributions (`TYPE=MIXTURE`).
  - Other models.
Exploratory Factor Analysis

Okay, this isn’t that advanced.

- Exploratory Factor Analysis is easy in that there is no need for a `MODEL` statement.
- It is our first use of the `ANALYSIS` command.
  - This analysis consists of an analysis type (EFA), which requires two arguments, the minimum and maximum number of factors to be extracted.
  - One can also add options, like `ROTATION` in this code:

```plaintext
VARIABLE:  NAMES = ID sex x1-x3 y1-y3 z1-z3;
           USEVAR = x1-z3;

ANALYSIS:  TYPE = EFA 1 4;
           ROTATION = PROMAX;
```
Mixed and Multilevel Modeling

Different conceptually, but often equivalent.

- There are a number of ways to deal with mixed and multilevel models in Mplus.
  - If you can specify them using the discrete-time SEM approach, more power to you.
- Mplus differentiates between growth curves (its primary mixed functionality) and multilevel models (in which it focuses on two-level models).
  - First, we’ll talk about a different way to deal with growth curves.
  - Then we’ll deal with the multilevel capabilities.
Mplus Basics Latent Variable Non-SEM Monte Carlo

Mixed & Growth Curve Models

- Mplus allows for the specification of latent growth curves and other random effect models using **ANALYSIS:** TYPE=RANDOM, using a "|" for model specification.
  - For those unfamiliar, this is Shift+\, which looks like a dashed line, pronounced “pipe.”
  - The focus is certainly on growth curves, but you can trick it a little bit.

- The general format is:

  growth parameters | manifest variables@time information
Mixed & Growth Curve Models

This is the standard linear growth curve:

\[ i \ s \ | \ x1@0 \ x2@0.25 \ x3@.5 \ x4@.75 \ x5@1; \]

- **TYPE=RANDOM** need not be specified, but is implied, by this.
- When two variables are on the left side of the pipe, a linear model is fit.
- The (latent) variables specified on the left side can take any name you give them; “i” and “s” are common through the Mplus manual.
Mixed & Growth Curve Models

- We can fit quadratic models:

```
MODEL:
i s q | x1@0 x2@0.25 x3@.5 x4@.75 x5@1;
```

- or cubic models:

```
MODEL:
i s q c | x1@0 x2@0.25 x3@.5 x4@.75 x5@1;
```

- The format is the same.
  - First variable is always intercept.
  - After that, increasing powers of the time variables.
Mixed & Growth Curve Models
What if time isn’t fixed?

▶ **TYPE=RANDOM** models can put a **TSCORES** option in the **VARIABLES** command.

▶ This option specifies variables that contain time information for mixed models that shouldn’t be in the covariance calculation.

▶ For example:

```
VARIABLE:
TSCORES = t1-t5;
MODEL:
i s q | x1-x5 AT t1-t5;
```

▶ This is a quadratic growth curve model for longitudinal data with individually varying times of observation.

▶ You can do more.
**TYPE=RANDOM tricks**

- As an aside, the pipe in `TYPE=RANDOM` is used to create new random effect variables, generally as the product of two other variables.

- This creates a random regression effect “rand”:

  ```
  rand | y ON x; !Requires TYPE=TWOLEVEL
  ```

- `XWITH` specifies multiplication, which is very handy for interactions between latent variables:

  ```
  int | i XWITH s;
  X2 | X XWITH X;
  ```
Multilevel Modeling

- Multilevel modeling (or hierarchical linear modeling, or random effect modeling, etc) is an important part of modern psychological research.

- In Mplus, this requires three components:
  - The identification of a CLUSTERing variable in the VARIABLE command,
  - The specification of TYPE=TWOLEVEL in the ANALYSIS command, and
  - Between and within group models in the MODEL command.

- Let’s see how it works!
Multilevel Modeling

VARIABLE:  NAMES = ID x y z
          WITHIN = x;
          BETWEEN = z;
          CLUSTER = ID;

ANALYSIS: TYPE=TWOLEVEL;

MODEL:

%WITHIN%
  y ON x;
%BETWEEN%
  y ON z;

!Defaults Shamelessly Employed
Multilevel Modeling

MODEL:

%WITHIN%

y ON x;

%BETWEEN%

y ON z;

▶ *y* is predicted from *x*, which varies at the individual level, and *z*, which varies at the cluster or group level.

▶ There are, of course, much more complex options.
  ▶ In particular, multilevel mixture models and higher order clustering.
  ▶ These topics are officially special topics, and are too in depth to be dealt with here.
Questions on Mixed/Random and Multilevel Models

- Depending on intent and specification, one could use these two styles of analysis for very similar problems.
- They both have strengths and drawbacks, and expect different data formats.
  - Random components expect wide-format one-entry-per-row data.
  - Multilevel is expects the tall-format presentation of clustered data.
- They’re very useful parts of Mplus.
Mixture Modeling

- Mixture modeling is supported under the `ANALYSIS` command’s `TYPE=MIXTURE` option.
- Required components:
  - A `VARIABLE` which designates the number and name of classes.
  - Some differences in the models across classes to make the model identifiable.
  - The overall model is stated with `%OVERALL%`, with the models for different classes referenced with `%classvariable#class%`.

```plaintext
VARIABLE: NAMES = ID sex x1-x3 y1-y3 z1-z3;
USEVAR = x1-z3;
CLASSES = c (2);

ANALYSIS: TYPE=MIXTURE;
```
Mixture Modeling

Here's a (relatively) simple example.

VARIABLE:  NAMES = ID sex x1-x3 y1-y3 z1-z3;
           USEVAR = x1 y1;
           CLASSES = c (2);

ANALYSIS:  TYPE=MIXTURE;

MODEL:

%OVERALL%
   y1 ON x1;!Regression
   y1 ON c;!Classes have different means
   c ON x1;!Logistic Regression
%c#2%
   y1 ON x1;!Regression depends on class
Here’s a growth mixture model, using the parameterization we discussed for TYPE=RANDOM.

VARIABLE: NAMES = ID sex x1-x3 y1-y3 z1-z3;
USEVAR = x1-x3;
CLASSES = c (2);

ANALYSIS: TYPE=MIXTURE;
STARTS = 20 2;
!20 sets of starting values
!2 sets used for final evaluation

MODEL:

%OVERALL%
i s | x1@0 x2@0.5 x3@1; !Growth Model
!Class Mean Differences
i s ON c;
Mixture Modeling

Good time for questions

- Mixture output includes:
  - Likelihood based fit.
  - Class membership, proportions and entropy.
  - Parameter estimates regarding binary latent class variable.

- Other features of mixture modeling framework:
  - Multiple sets of classes may be used.
  - Known classes may be used, either for mixture capabilities or an alternative approach to multiple group models.
  - Multilevel mixture modeling & mixture modeling for complex survey data.
Other Models

- Survival Analysis.
  - Parametric and Cox PH survival models are options.
  - ANALYSIS: BASEHAZARD=ON or OFF

- Parameterization:
  - VARIABLE command takes SURVIVAL and TIMECENSORED statements to identify time and censoring variables.
  - The variable referenced by SURVIVAL is used in the model, with the TIMECENSORED associated with in the appropriate way.

- Questions?
Monte Carlo Capabilities

- Mplus can be used for Monte Carlo studies.
  - Of course, you could do this manually, using another program to generate data and a batch file to call Mplus.
- The real virtue of Mplus in this regard is that it can generate data for Monte Carlo work based on your parameters.
- Scripts for Monte Carlo studies require the following commands:
  - `MONTECARLO`, which replaces the `DATA` and `VARIABLE` commands to describe the dimensions of the data, as well as the simulation parameters.
  - `MODEL POPULATION`, which specifies the underlying population model which is used to generate the data.
  - `MODEL MISSING` (optional), which specifies a model for missingness.
  - `MODEL`, which specifies the model you’re trying to fit.
Monte Carlo Capabilities

The MONTECARLO Command

The following statements fall under the MONTECARLO command:

- **NAMES**, which names the variables;
- **NOBSERVATIONS**, which states the number of observations per dataset;
- **NREPS**, which gives the number of repetitions for the simulation;
- **SEED**, which specifies the random seed (optional);
- **CUTPOINTS**, which specifies some variables as binary and gives a cutpoint for the standard normal distribution used in data generation.
- **PATMISS**, which gives pattern(s) of missing data. If multiple patterns are used, they are separated by a | and the PATPROBS statement describes the probabilities of each pattern.
Monte Carlo Capabilities
The MONTECARLO Command

MONTECARLO:
  NAMES = x1-x5 sex;
  NOBSERVATIONS = 100;
  NREPS = 500;
  SEED = 090219;
  CUTPOINTS = sex (0);
  PATMISS = x1 (.1) x2 (.05) x3 (.2) x4 (.1) |
           x1 (.5) x2 (.5) x3 (.05) x4 (.3);
  PATPROBS = .9 | .1;
Monte Carlo Capabilities

The **MODEL POPULATION** Command

```plaintext
MODEL POPULATION:
level BY x1-x5@1;
slope BY x1@0;
slope BY x2*.25;
slope BY x3*.5;
slope BY x4*.75;
slope BY x5@1;
level; slope; level WITH slope*.2;
[x1-x5@0; level; slope;]
level ON sex*1;
slope ON sex*0;
```

▶ Why use @ and * interchangeably?
Monte Carlo Capabilities

The `MODEL` Command

```plaintext
MODEL:
  level BY x1-x5@1;
  slope BY x1@0;
  slope BY x2*.25;
  slope BY x3*.5;
  slope BY x4*.75;
  slope BY x5@1;
  level; slope; level WITH slope*.2;
  [x1-x5@0; level; slope;]
  level ON sex*1;
  slope ON sex*0;

▶ If you’re fitting the same model you’re generating.
```
Monte Carlo Capabilities

Advanced Usage

Mplus’ Monte Carlo Capabilities can be used to generate data for and fit any of the models we’ve discussed here today.

It also has the following capabilities:

- Output generated data for later analysis.
- Read previously generated data for a Monte Carlo study.
- Output estimates as a data file for later analysis.

There’s not a lot I can say that the manual doesn’t on these topics.
Summary

- There are many topics we didn’t cover.
  - Technical output control, from viewing the (LISREL style) matrices to modification indices.
  - Data and statistical output for later analysis.
  - The `PLOT` command.
  - Support for advanced data structures, from long-to-wide transformations to cohort data.
  - Bootstrapping.
  - Categorical Latent Variables and their scale factors.
  - Indirect effects and their standard errors.
  - Integration options (`INTEGRATION=MONTECARLO` is good to know).

- Mplus is a good tool. It’s not a perfect tool. Use it, don’t use it, and/or write a competing program.
Acknowledgements

- John Nesselroade.
- Jack McArdle, Aki Hamagami, Kevin Grimm.
- All members of the OpenMx development team.