Supplemental MPLUS Material for *Longitudinal Analysis: Modeling Within-Person Fluctuation and Change* by Lesa Hoffman (2015, Routledge Taylor & Francis)

General Notes about MPLUS Syntax (see also http://www.statmodel.com/index.shtml)

- **Basic rules about MPLUS syntax and output:**
  - MPLUS has only two kinds of files, both of which are just text files (so you can open and edit them in notepad or Word): **.inp** is for input (syntax), and **.out** is for output (the input syntax will re-appear at the top).
  - MPLUS is not case-sensitive or space-sensitive, but it will not read past 90 characters in each syntax line (check the column # in bottom right corner of the screen to make sure you don't go past 90).
  - MPLUS primary commands are `BLUE` and end in **colons**; subcommands are `BLACK` and end in **semi-colons**.
  - Comments are indicated with `!` at the beginning of each so that the text that follows will turn `GREEN`. As of v. 7.2, several lines can be commented out by starting the first with `!*` and ending the last line with `*!`
  - There is no way of selecting just parts of code to run—the entire input file is read each time. So if you want to only run certain parts of the syntax, you can comment out the irrelevant lines of syntax via the `!` for each line.
  - The terms `IS`, `ARE`, and `=` in defining subcommands (see below) are interchangeable.
  - You can abbreviate variables using the `ALL` keyword or using a dash for contiguous names that *end* in ordered numbers (e.g., `var1-var10` will abbreviate `var1`, `var2`, `var3`…. `var10`, but `var1x-var10x` will not work).

- **Getting data into MPLUS:**
  - There is no provision for viewing data within MPLUS. For this reason, although it is possible to perform variable transformations within MPLUS (e.g., centering, stacking/unstacking data), it will be easiest to make sure this is done correctly if you do so in the native file (e.g., SPSS, SAS, STATA) in which you can see the data.
  - You will need to write in MPLUS syntax the names of ALL variables in the dataset, so it will be more convenient if you limit your MPLUS dataset to just those variables you will need for your analyses, making especially sure to remove any string variables or date variables.
  - **All variable names and parameter labels must use 8 characters or fewer**. MPLUS does not know what the variables were called within the original data, so you can call them anything 8 characters or fewer in MPLUS.
  - Beware of missing data! Although MPLUS accepts `BLANK` as a missing data indicator, this may not work as well as a defined missing data code (e.g., −9999). It will be easiest if all variables have the same missing data code.
  - MPLUS only accepts tab-delimited files (**.dat**), fixed-format text files (**.dat**), or comma-separated-values files (**.csv**). Of these, **.csv** is most convenient because it opens by default in Excel so that you can view it as needed. If importing a delimited file fails, you can try the other formats, with fixed-format as the last resort.
  - Here is how to save data from other programs to a **.csv** format for use in MPLUS:
    - **Through SPSS windows:** FILE → SAVE AS → change "Save as type" box to "comma delimited (*.csv)" and UNCHECK the box that says "Write variable names to spreadsheet". MPLUS does NOT read variable names from the file, and will give you an error message if they appear on the first line instead of data.
    - **Using SPSS syntax `SAVE TRANSLATE` below:** `OUTFILE` tells is which SPSS file to export, `/TYPE` indicates a CSV format, `/REPLACE` means it will be replaced if a file already exists with that name, and

`/CELLS=VALUES` will save actual data (numbers) instead of any value labels included.

```
SAVE TRANSLATE OUTFILE="F:\folder\mydata.csv" /TYPE=CSV /REPLACE /CELLS=VALUES.
```

- **Using SAS syntax `PROC EXPORT` below:** `DATA` tells it which SAS file to export, `OUTFILE` lists the path and name of the new .csv file, `REPLACE` means it will be replaced if a file already exists with that name, and `PUTNAMES=NO` tells it not to write the names to the top of the .csv file.

```
PROC EXPORT DATA=work.mydata OUTFILE="F:\folder\mydata.csv"
            DBMS=CSV REPLACE; PUTNAMES=NO; RUN;
```

- **Using STATA syntax `export delimited` below:** `using` lists the path and name of the new .csv file, `replace` means it will be replaced if a file already exists with that name, `delimiter` indicates a comma-delimited file, `nolabel` will save actual data (numbers) instead of any value labels included, and `novarnames` tells it not to write the names to the top of the .csv file.

```
export delimited using "F:\folder\mydata.csv", ///
                 delimiter(",") replace nolabel novarnames
```

- When you begin an MPLUS analysis, the VERY FIRST THING YOU SHOULD DO is make sure your data were read correctly. To do so, use `TYPE = BASIC` as the `ANALYSIS:` subcommand or `SAMPSTAT` as an `OUTPUT:` subcommand. These will provide means, variances, and covariances for your continuous variables, or frequency tables for any variables that are listed as `CATEGORICAL =` instead. These should match those of your original data exactly (unless you have missing data). But if they are not even close, then chances are it is not reading your data correctly yet. Verify that you have listed the variable names in the exact order in which they appear in the data and that you didn't forget to list any variables (e.g., filter variables that were appended to the end of the file).

- **MPLUS primary commands:**
  - `TITLE:` This is optional and will print at the top of your output if included.
  - `DATA:` This is NOT optional and is how you tell MPLUS where your data are and their format. Its subcommands:
    - `FILE =` gives the name of the data file. The path to the file is also needed if the syntax is not stored in the same location as the data file:
      - `FILE = mydata.csv`; or `FILE = C:\SomeOtherFolder\mydata.csv;`
    - `FORMAT =` tells it your data format, such as delimited (default free format: `FORMAT = free`) or fixed-format
      - If you have fixed-format data, you need to give its formatting statement. For example, if you had one ID variable with 4 characters and no decimals, followed by 24 variables with 4 characters, 2 of which were decimals: `FORMAT = F4.0 24F4.2;`
    - `TYPE =` tells it whether you have individual data (the default) or summary data (e.g., covariance matrix)
      - `TYPE = individual;` or `TYPE = fullcov;`
  - There are two data transformation commands (that appear in black) that allow you to restructure data by unit (although they are not available for some models, such as three-level models).
    - `DATA LONGTOWIDE:` transposes per unit from rows to columns (i.e., from stacked to multivariate), in which `LONG` lists the stacked variables to be transposed (each separated by `|`), `WIDE` lists the new names of the columns (each separated by `|`), `IDVARIABLE` lists the variable that links rows from the same unit, and `REPETITION` lists the variable that indexes the specific rows that should go into each column (e.g., `time`) along with the range of its values. For example:

```
DATA LONGTOWIDE:
  LONG = xvar | yvar;
  WIDE = xvar1-xvar5 | yvar1-yvar5;
  IDVARIABLE = L2IDname;
  REPETITION = time (1-5);
```

- **`DATA WIDETOLONG:`** transposes per unit from columns to rows (i.e., from multivariate to stacked) using the same options as in **`DATA LONGTOWIDE:`** above. For example:

```
DATA WIDETOLONG:
  WIDE = xvar1-xvar5 | yvar1-yvar5;
  LONG = xvar | yvar;
  IDVARIABLE = L2IDname;
  REPETITION = time (1-5);
```

- **`DEFINE:`** This is optional, and is how you make new variables out of existing variables. You then need to list any newly created variables on the **`USEVARIABLES =`** list AFTER the existing variables. Note that the **`DEFINE:`** command is executed before the **`DATALONGTOWIDE:`** or **`DATAWIDETOLONG:`** commands.
  - For example, to create a new variable **`c_pred`** centered at 30 from the original variable **`pred`**:
    - **`DEFINE: c_pred = pred – 30;`**
  - To recode a predictor, **`EQ`** is used to evaluate an equality, but **`=`** is used to transform the new variable:

    -
      ```
      DEFINE: IF oldvar EQ 1 THEN newvar=0;
              IF oldvar EQ 2 THEN newvar=1;
      ```

- **`VARIABLE:`** This is NOT optional and is how you tell MPLUS what your variables are to be called, as well as which ones you are using for what purpose via the subcommands below:
  - **`NAMES =`** is where you list every variables in the original dataset (i.e., before any **`DEFINE:`**, **`DATA LONGTOWIDE:`**, or **`DATA WIDETOLONG:`** commands are executed), terminated by a semi-colon
  - **`USEVARIABLES =`** is where you list just the variables in your **`MODEL:`**, such that variables in the original dataset are listed first, followed by any new variables created using **`DEFINE:`**, terminated by a semi-colon
  - **`USEOBSERVATIONS =`** is optional and is how you tell MPLUS to use only certain cases, for example:
    - **`USEOBSERVATIONS = gender EQ 1;`**
  - **`MISSING =`** is optional and is how you tell MPLUS how missing data are indicated, for example:
    - **`MISSING = ALL (-9999);`**
  - **`IDVARIABLE =`** is optional and tells MPLUS what variable indicates which case is which—if you add this option, then any outputted dataset you ask for will have this variable included, which is necessary if you want to merge any outputted data from MPLUS into the original data
    - **`IDVARIABLE = IDname;`**
  - **`AUXILIARY =`** is how you tell MPLUS to keep the other variables not currently in the model in any outputted datasets, or to use those variables in the missingness model
    - **`AUXILIARY = gender age group;`**
  - See the MPLUS manual for further suboptions that indicate non-normal distributions for outcome variables, such as **`CATEGORICAL =`**, **`COUNT =`**, and **`CENSORED =`** (each terminated by a semi-colon)

- The following subcommands are used for multilevel models in MPLUS:
  - **`CLUSTER =`** tells MPLUS the ID variables by which higher-level units are organized

    For example, in a two-level model:
    ```
    CLUSTER = L2IDname;
    ```
    In a three-level model, the ID variable for the highest level 3 is listed first, followed by the ID variable for level 2. MPLUS assumes that any repeated values of the level-2 ID variable are still unique cases (e.g., values for L2IDname can be 1….10 within each L3IDname). For example:
    ```
    CLUSTER = L3IDname L2IDname;
    ```
  - **`WITHIN =`** is used to identify observed predictor variables to be included ONLY at level 1

    For example, two level-1 predictor variables in a two-level model:
    ```
    WITHIN = L1x1 L1x2;
    ```
  - **`BETWEEN =`** is used to identify observed predictor variables to be included ONLY at level 2 or level 3

    For example, two level-2 predictor variables in a two-level model:
    ```
    BETWEEN = L2x1 L2x2;
    ```
    In a three-level model, you also have to distinguish which variables belong to level 2 versus level 3.

    For example, two level-2 predictor variables only:
    ```
    BETWEEN = (L2IDname) L2x1 L2x2;
    ```
    For example, two level-3 predictor variables only:
    ```
    BETWEEN = (L3IDname) L3x1 L3x2;
    ```
    For example, two level-2 predictor variables AND two level-3 predictor variables:
    ```
    BETWEEN = (L2IDname) L2x1 L2x2 (L3IDname) L3x1 L3x2;
    ```
  - If a variable is measured at level 1 and has variance at multiple levels for which you want to estimate level-1 residual and level-2 and/or level-3 random intercept variances (i.e., it is being modeled as if it were an outcome), then do NOT include it on the **`WITHIN =`** or **`BETWEEN =`** lists.

- **`ANALYSIS:`** This is not usually optional, and is how you set any estimation options. These will be model-specific and include subcommands for **`TYPE =`**, **`ESTIMATOR =`**, and others as needed, each terminated by a semi-colon.
  - Here's how you get the descriptive statistics to check your data: **`TYPE = BASIC;`**
  - To estimate two-level models: **`TYPE = TWOLEVEL RANDOM;`**
  - To estimate three-level models: **`TYPE = THREELEVEL RANDOM;`**
  - To request full-information maximum likelihood: **`ESTIMATOR = ML;`**
  - To request full-information robust maximum likelihood: **`ESTIMATOR = MLR;`**
  - Note: There is currently no option for full-information restricted maximum likelihood (as of v. 7.3)
  - For computers with multiple processors, you can tell it how many to use (e.g., 4): **`PROCESSORS = 4;`**

- **`OUTPUT:`** These subcommands request specific pieces of output not provided by default, such as standardized coefficients, analysis of residuals, and sample statistics for checking the data. This will vary by model.
  - To get fully standardized coefficients: **`STDYX;`**
  - To get residual diagnostics (e.g., for path models or structural equation models): **`RESIDUAL;`**

  - To get modification indices (e.g., for path models or structural equation models), in which the critical value for the improvement to the model chi-square is given in parentheses: **`MODINDICES (3.84);`**

- **`SAVEDATA:`** This is how you ask for outputted datasets, such as factor scores or model parameters.
- **`PLOT:`** There are many kinds of plots available in MPLUS. To get all of them you could ever get for any model:
  - **`TYPE = PLOT1 PLOT2 PLOT3;`**

- **`MODEL:`** The syntax in this section is always model-specific and uses just a few different key words:
  - **`BY`** is used to define latent variables, for example: **`factor BY var1-var10;`**
  - **`WITH`** is to define covariances, for example: **`var1 WITH var2;`**
  - All possible covariances among a set of variables can be defined within a single statement by listing all of them on both sides of the **`WITH`**, for example: **`var1 var2 var3 var4 WITH var1 var2 var3 var4;`**
  - **`PWITH`** can be used for pairwise covariances, such as between **`x1`** and **`y1`**, **`x2`** and **`y2`**, and so on: **`x1 x2 x3 x4 PWITH y1 y2 y3 y4;`**
  - **`ON`** is used to define regression slopes, for example: **`y1 ON x1;`**
  - Multiple predictors and/or multiple outcomes can be designated with a single **`ON`** statement. For example, to have both **`y1`** and **`y2`** predicted by both **`x1`** and **`x2`** (four regressions in total): **`y1 y2 ON x1 x2;`**
  - **`PON`** is used to define pairwise regression slopes. For example, to have just **`x1`** predict **`y1`** and just **`x2`** predict **`y2`** (two regressions in total): **`y1 y2 PON x1 x2;`** is the same as: **`y1 ON x1; y2 ON x2;`**
  - Although **`DEFINE:`** is used to create interaction terms between observed predictor variables at the same level of analysis, **`XWITH`** is used instead to specify interactions involving any latent variables:
    - Interaction called **`F1F2int`** between two latent variables **`F1`** and **`F2`**: **`F1F2int | F1 XWITH F2;`**
    - Interaction called **`F1F2int`** between latent **`F1`** and observed **`x1`**: **`F1x1int | F1 XWITH x1;`**
  - MPLUS infers based on your model syntax whether each variable is a predictor or an outcome and estimates its model parameters accordingly. In terms of labeled output, variables in the likelihood that are only predictors will have estimated "means", "variances", and "covariances" (marginal statistics). Variables that are outcomes (even if they are also predictors) will have estimated "intercepts" (expected mean when all predictors = 0), "residual variances" (its variance leftover that is unpredicted), and "residual covariances" (covariance between residuals specifically, not the original variables). Either way, the syntax will be the same to refer to means or intercepts, to variances or residual variances, and to covariances or residual covariances.
  - Predictor variables are NOT included in the likelihood by default, whereas outcome variables are included in the likelihood by default. Any variable in the likelihood has distributional assumptions (multivariate normal unless specified otherwise) and can have missing data assuming missing at random.
  - For example, specifying a linear regression of **`x1`** and **`x2`** predicting **`y1`** estimates these parameters:
    - Regression slopes for **`x1`** and **`x2`** predicting **`y1`** using **`ON`**: **`y1 ON x1 x2;`**
    - By default: Fixed intercept for **`y1`**, referred to using **`[ ]`**: **`[y1];`**
    - By default: Residual variance for **`y1`**, referred to by listing its name: **`y1;`**

- By default, no means, variances, or covariances would be estimated for `x1` and `x2`, which means they are not in the likelihood, and thus have no distributional assumptions and cannot have missing data. However, to bring them into the likelihood (i.e., so that they can have missing at random data along with distributional assumptions), refer to their means, variances, or covariances as estimated parameters:
  - Means for `x1` and `x2`, referred to using `[ ]`: `[x1 x2];`
  - Variances for `x1` and `x2`, referred to by listing their names: `x1 x2;`
  - Covariance between `x1` and `x2`, referred to using `WITH`: `x1 WITH x2;`
- Free (estimated) parameters are indicated with a `*` and fixed parameters with `@`. In many cases the default is free, so `*` is not usually necessary to write (but `@` is necessary when it is not the default, as in most cases).
  - So `[x1 x2];` is the same as `[x1* x2*];` and `x1 WITH x2;` is the same as `x1 WITH x2*;`
    To force a covariance to be 0 instead: `x1 WITH x2@0;`

Understanding the MPLUS MODEL Commands used for Longitudinal and Multilevel Modeling

- **Defining levels of analysis**
  - In a two-level or three-level model, the level-1 model begins with keyword `%WITHIN%`
  - In a two-level model, the level-2 model begins with keyword `%BETWEEN%`
  - In a three-level model, levels 2 and 3 are differentiated using a `%BETWEEN%` keyword that includes the level-2 and level-3 ID variables that were indicated using the `CLUSTER =` subcommand in the `VARIABLE:` command
    - To begin the level-2 model: `%BETWEEN L2IDname%`
    - To begin the level-3 model: `%BETWEEN L3IDname%`
  - Variables not explicitly listed on the `WITHIN =` or `BETWEEN =` subcommands within the `VARIABLE:` command are treated as outcomes. Their variances are estimated by default at each level—these are labeled as either “variance” if the variable is NOT predicted within the model, or “residual variance” if the variable IS predicted within the model. Either way, the variance or residual variance is referred to by listing the name of the variable. However, the intercept or mean for the outcome (“intercept” if predicted; “mean” if not predicted) is specified only at the highest level of the model. Below is how to refer to these estimated parameters.

  - For example, an empty two-level model for outcome `y1` would include the following:
    - Begin level-1 model: `%WITHIN%`
    - By default, an estimated level-1 (“residual”) variance: `y1;`
    - Begin level-2 model: `%BETWEEN%`
    - By default, an estimated level-2 (“random intercept”) variance: `y1;`
    - By default, an estimated fixed intercept at level 2: `[y1];`
    - To estimate a single-level (“e-only”) model, remove the level-2 variance: `y1@0;`

- For example, an empty three-level model for outcome `y1` would include the following:
  - Begin level-1 model: `%WITHIN%`
  - By default, an estimated level-1 ("residual") variance: `y1;`
  - Begin level-2 model: `%BETWEEN L2IDname%`
  - By default, an estimated level-2 ("random intercept") variance: `y1;`
  - Begin level-3 model: `%BETWEEN L3IDname%`
  - By default, an estimated level-3 ("random intercept") variance: `y1;`
  - By default, an estimated fixed intercept is now at level 3: `[y1];`
  - To estimate a two-level model instead, remove the level-3 variance: `y1@0;`
  - To estimate a single-level ("e-only") model instead, under `%BETWEEN L2IDname%` add `y1@0;`
- In general, removing a variance at a given level involves referring to the variable with `@0` after it.

- **Specifying fixed and random effects of level-1 predictors in two-level or three-level models**
  - The following applies to both main effects of level-1 predictors and to interactions among level-1 predictors (the latter must be represented explicitly using new separate variables).
  - A level-1 variable will be used as an observed level-1 predictor if it is included on the `WITHIN =` subcommand of the `VARIABLE:` command, which means it is not included in the likelihood, it has no distributional assumptions, and any cases with missing values will be listwise-deleted. As a second option, a level-1 predictor listed on the `WITHIN =` subcommand whose mean, variance, or covariance is referred to explicitly in the syntax will be brought into the likelihood at level 1 only. This means that its variance will be estimated (although it will not be partitioned across all model levels), so distributional assumptions still apply and its values can be missing at random without listwise deletion. As a third option, if not listed on the `WITHIN =` subcommand, the variance of a level-1 variable will be estimated at each level as part of the likelihood, it will have distributional assumptions, and its values can be missing at random without listwise deletion. However, adding effects of a level-1 variable proceeds similarly in any of these three cases, as described next.
  - There are two ways to add the effect of a level-1 predictor that result in different parameters estimated by default: (a) directly using the keyword `ON`, or (b) indirectly by creating a new "placeholder" variable. If you want to estimate its random effect at a higher level and/or a cross-level interaction involving the level-1 predictor and a higher-level predictor, you MUST use the indirect placeholder method (b).
  - In either a two-level or three-level model, use of <u>direct method (a)</u> by default results in ONLY a level-1 fixed effect (reported within the level-1 model).
  - For example using <u>direct method (a)</u>, to add a level-1 fixed effect predicting `y1` from level-1 predictor `L1x`:
    - Under the `%WITHIN%` level-1 model, a fixed effect is added: `y1 ON L1x;`
  - In a two-level model, use of <u>placeholder method (b)</u> results by default in a level-1 fixed effect and a corresponding level-2 random effect variance (both reported within the level-2 model output). No random effect covariances are added by default, so you will have to add those manually using a `WITH` statement at each level, as shown below.
  - For example using <u>placeholder method (b)</u>, to add a fixed or random effect of level-1 predictor `L1x`:
    - Under the `%WITHIN%` level-1 model, a new placeholder variable of 8 or fewer characters is first defined using the `|` as shown (here, called `L1xslope`): `L1xslope | y1 ON L1x;`

- Under the **`%BETWEEN%`** level-2 model, the level-1 placeholder variable will have a mean and variance estimated by default, which correspond to its fixed effect and its random effect variance:

To refer to its estimated fixed effect:	**`[L1xslope];`**
To remove its fixed effect:	**`[L1xslope@0];`**
To refer to its estimated level-2 random effect variance:	**`L1xslope;`**
To add its covariance with the level-2 random intercept:	**`y1 WITH L1xslope;`**
To remove its level-2 random effect variance:	**`L1xslope@0;`**

- In a three-level model, use of placeholder method (b) by default results in a level-1 fixed effect (reported within the level-3 model), as well as corresponding level-2 and level-3 random effect variances. By default no covariances with other random effects are added at either level 2 or 3, so you will have to add those manually using a **`WITH`** statement at each level in which its random effect variance is estimated.
- For example using placeholder method (b), to add a fixed or random effect of level-1 predictor **`L1x`**:
  - Under the **`%WITHIN%`** level-1 model, a new level-1 placeholder variable of 8 or fewer characters is first defined using the **`|`** as shown (here, called **`L1xslope`**): **`L1xslope | y1 ON L1x;`**
  - Under the **`%BETWEEN L2IDname%`** level-2 model, the level-1 placeholder variable will have a variance estimated by default, which corresponds to its random effect variance across level-2 units:

To refer to its estimated level-2 random effect variance:	**`L1xslope;`**
To add its covariance with the level-2 random intercept:	**`y1 WITH L1xslope;`**
To remove its level-2 random effect variance:	**`L1xslope@0;`**

  - Under the **`%BETWEEN L3IDname%`** level-3 model, the level-1 placeholder variable will have a mean and variance estimated by default, which correspond to its fixed effect and its random effect variance across level-3 units:

To refer to its estimated fixed effect:	**`[L1xslope];`**
To remove its fixed effect:	**`[L1xslope@0];`**
To refer to its estimated level-3 random effect variance:	**`L1xslope;`**
To add its covariance with the level-3 random intercept:	**`y1 WITH L1xslope;`**
To remove its level-3 random effect variance:	**`L1xslope@0;`**

- When using placeholder method (b), the fixed and random effects of the level-1 predictor will be labeled on the output as “intercept/mean” or “residual variance/variance”, respectively, depending on whether the level-1 predictor is part of any cross-level interactions with higher-level predictors:
  - If no cross-level interactions involving the level-1 predictor are estimated, its fixed effect will be reported as a “mean” (at level 2 in a two-level model or at level 3 in a three-level model) and its random effect variance(s) will be reported as a “variance” at each corresponding level.
  - Alternatively, if the level-1 predictor is part of a cross-level interaction with either a level-2 or level-3 predictor (whose specification is explained in more detail below), then its fixed effect will be reported as an “intercept” (i.e., as the expected fixed effect when any interacting predictors = 0) at level 2 in a two-level model or at level 3 in a three-level model.

- If the level-1 predictor is part of a cross-level interaction with a level-2 predictor, its level-2 random effect variance will be reported at level 2 as a "residual variance" (i.e., as variance that remains after being predicted by any cross-level interactions with level-2 predictors) or as a "variance" if not.
- If the level-1 predictor is part of a cross-level interaction with a level-3 predictor, its level-3 random effect variance will be reported at level 3 as a "residual variance" (i.e., as variance that remains after being predicted by any cross-level interactions with those level-3 predictors) or as a "variance" if not.

- **Specifying cross-level interactions involving level-1 predictors and level-2 or level-3 predictors**
  - Cross-level interactions of level-1 predictors with higher-level predictors require that its level-1 effect be specified using placeholder method (b), regardless of whether its random effect variances are also estimated:
    - Placeholder method (b): Under the `%WITHIN%` level-1 model, a new level-1 placeholder variable of 8 or fewer characters is first defined (here, called `L1xslope`): `L1xslope | y1 ON L1x;`
  - In a two-level model, its cross-level interaction with a level-2 predictor `L2x` (reported in the level-2 output) would then be added under the `%BETWEEN%` level-2 model directly: `L1xslope ON L2x;`
  - In a three-level model, its cross-level interaction with a level-3 predictor `L3x` (reported in the level-3 output) would then be added under the `%BETWEEN L3IDname%` level-3 model directly: `L1xslope ON L3x;`
  - In a three-level model, its cross-level interaction with a level-2 predictor `L2x` (reported in the level-2 output) CAN be added under the `%BETWEEN L2IDname%` level-2 model directly: `L1xslope ON L2x;`
  - However, if the cross-level interaction between the level-1 and level-2 predictor is part of an interaction with a level-3 predictor (i.e., it is part of a three-way interaction) and/or has a level-3 random effect variance, you must use placeholder method (b) syntax to specify the cross-level interaction between the level-1 and level-2 predictor instead:
    - Under the `%BETWEEN L2IDname%` level-2 model, a new level-2 placeholder variable of 8 or fewer characters is first defined (here, called `L1xL2x`): `L1xL2x | L1xslope ON L2x;`
    - Under the `%BETWEEN L3IDname%` level-3 model, the level-2 placeholder variable will have a mean and variance estimated by default, which correspond to its fixed effect and its random effect variance across level-3 units:

To refer to its estimated fixed effect:	`[L1xL2x];`
To remove its fixed effect:	`[L1xL2x@0];`
To refer to its estimated level-3 random effect variance:	`L1xL2x;`
To add its covariance with the level-3 random intercept:	`y1 WITH L1xL2x;`
To add its covariance with the level-3 random effect of `L1x`:	`L1xslope WITH L1xL2x;`
To remove its level-3 random effect variance:	`L1xL2x@0;`

- **Interpreting fixed effects of level-1 predictors in two-level or three-level models**
  - The following logic applies to any effect of a level-1 predictor (i.e., a main effect, an interaction with another level-1 predictor, or a cross-level interaction with higher-level predictors).
  - What the effect of a level-1 predictor actually means depends on two things: (1) how the level-1 variable is modeled (i.e., as an observed predictor or as a level-1 outcome whose variances at each level are estimated as part of the likelihood), and (2) how its level-1 effect is specified in the syntax.

- For a level-1 predictor whose variance at each level is NOT partitioned by the model: when specified using either direct method (a) or placeholder method (b), the interpretation of the level-1 predictor's fixed effect is determined by the variance it contains, the same as in other programs (and as described in chapter 8).
  - If the level-1 predictor has been variable-centered (i.e., its level-2 variance has been removed), its level-1 effect will be the "pure" level-1 within effect.
  - If the level-1 predictor has been left uncentered or centered at a constant (i.e., grand-mean-centering in which its level-2 and/or level-3 variance is still present), its level-1 effect will be a smushed effect UNLESS any variance it has at higher levels is included through model predictors for contextual effects (to be included at each level for which the level-1 predictor has variance), in which case the level-1 effect of the level-1 predictor is then its "pure" level-1 within effect as intended.
- For a level-1 predictor whose variance at each level IS being partitioned by the model, references to the variable at each level actually refer to its partitioned level-specific variance. Consequently, the level-1 predictor's effect is specifically for its level-1 variance as the predictor, such that it carries only the "pure" level-1 within effect. However, the interpretation of its level-2 effect (and of its level-3 effect in a three-level model) depends on how the level-1 effect was specified, using method (a) or method (b):
  - If using direct method (a), its level-2 effect is the total level-2 between effect (i.e., not controlling for the absolute level-1 predictor value). Likewise, in a three-level model, its level-3 effect is the total level-3 between effect (i.e., not controlling for the absolute values of the predictor at levels 1 or 2).
  - In contrast, when using placeholder method (b), its level-2 effect is a contextual level-2 (incremental between) effect instead (i.e., after controlling for the absolute level-1 predictor value). Likewise, in a three-level model, its level-3 effect is a contextual level-3 (incremental between) effect instead (i.e., after controlling for the absolute values of the predictor at levels 1 and 2).This is the case regardless of whether its level-2 and/or level-3 random effect variance—differentially allowed by use of the method (b) placeholder syntax for the level-1 effect—is actually estimated in the model.

- **Specifying fixed and random effects of level-2 predictors in two-level or three-level models**
  - The following applies to both main effects of level-2 predictors and to interactions among level-2 predictors (the latter must be represented explicitly using new separate variables).
  - A level-2 variable will be used as an observed level-2 predictor if it is included on the **`BETWEEN =`** subcommand of the **`VARIABLE:`** command, in which case it is not included in the likelihood, it has no distributional assumptions, and any cases with missing values will be listwise-deleted. As a second option, a level-2 predictor listed on the **`BETWEEN =`** subcommand whose mean, variance, or covariance is referred to explicitly in the syntax will be brought into the likelihood at level 2 only. This means that its variance will be estimated (although it will not be partitioned across levels 2 and 3), so distributional assumptions still apply and its values can be missing at random without listwise deletion. As a third option, if not listed on the **`BETWEEN =`** subcommand, the variance of a level-2 variable will be estimated at levels 2 and 3 as part of the likelihood, it will have distributional assumptions, and its values can be missing at random without listwise deletion. However, adding effects of a level-2 variable proceeds similarly in any of these three cases, as described next.
  - As with level-1 predictors, there are two ways to add the effect of a level-2 predictor that result in different parameters estimated by default: (a) directly using the keyword **`ON`**, or (b) indirectly by creating a new "placeholder" variable. Two-level models must use direct method (a). Three-level models may use either method, but if you want to estimate the level-2 predictor's random effect at level 3 or a cross-level interaction involving the level-2 predictor and a level-3 predictor, you MUST use method (b).
  - In either a two-level or three-level model, use of direct method (a) by default results in ONLY a level-2 fixed effect (reported within the level-2 model).

- For example using direct method (a), to add a level-2 fixed effect predicting **`y1`** from level-2 predictor **`L2x`**:
  - In a two-level model, under the **`%BETWEEN%`** level-2 model, a fixed effect is added: **`y1 ON L2x;`**
  - In a three-level model, under the **`%BETWEEN L2IDname%`** level-2 model, a fixed effect is added: **`y1 ON L2x;`**
- For use in three-level models only, the placeholder method (b) by default results in a level-2 fixed effect and a corresponding level-3 random effect variance (both reported within the level-3 model output). By default no covariances with other level-3 random effect are added, so you will have to add those manually using a **`WITH`** statement in the level-3 model, as shown below.
- For example using placeholder method (b), to add a fixed or random effect of level-2 predictor **`L2x`**:
  - Under the **`%BETWEEN L2IDname%`** level-2 model, a new placeholder variable of 8 or fewer characters is first defined using the **`|`** as shown (here, called **`L2xslope`**): **`L2xslope | y1 ON L2x;`**
  - Under the **`%BETWEEN L3IDname%`** level-3 model, the level-2 placeholder variable will have a mean and variance estimated by default, which correspond to its fixed effect and its random effect variance:

To refer to its estimated fixed effect:	**`[L2xslope];`**
To remove its fixed effect:	**`[L2xslope@0];`**
To refer to its estimated level-3 random effect variance:	**`L2xslope;`**
To add its covariance with the level-3 random intercept:	**`y1 WITH L2xslope;`**
To remove its level-3 random effect variance:	**`L2xslope@0;`**

- When using placeholder method (b), the fixed and random effects of the level-2 predictor will be labeled on the output as “intercept/mean” or “residual variance/variance”, respectively, depending on whether the level-2 predictor is part of any cross-level interactions with level-3 predictors:
  - If no cross-level interactions involving the level-2 predictor and a level-3 predictor are estimated, its fixed effect will be reported as a “mean” at level 3 in a three-level model and its random effect variance will be reported as a “variance” at level 3.
  - In contrast, if the level-2 predictor is part of a cross-level interaction with a level-3 predictor (whose specification is explained in more detail below), then its fixed effect will be reported as an “intercept” (i.e., as the expected fixed effect when any interacting predictors = 0) at level 3 and its random effect variance will be reported at level 3 as a “residual variance” (i.e., as variance that remains after being predicted by any cross-level interactions with those level-3 predictors).

- **Specifying cross-level interactions involving level-2 predictors with level-3 predictors in three-level models**
  - A cross-level interaction of a level-2 predictor with a level-3 predictor requires that the level-2 effect be specified using placeholder method (b), regardless of whether its level-3 random effect variance is estimated:
    - Placeholder method (b): Under the **`%BETWEEN L2IDname%`** level-2 model, a new level-2 placeholder variable of 8 or fewer characters is first defined (here, called **`L2xslope`**): **`L2xslope | y1 ON L2x;`**
  - Its cross-level interaction with a level-3 predictor **`L3x`** (reported in the level-3 output) would then be added under the **`%BETWEEN L3IDname%`** level-3 model directly: **`L2xslope ON L3x;`**

- **Interpreting fixed effects of level-2 predictors in two-level or three-level models**
  - In a two-level model, level-2 predictors only contain level-2 variance, and thus can only have level-2 effects (i.e., you do not have to worry about smushing them). However, the effect of a level-2 variable with a counterpart at level 1 will have a different interpretation depending on the variance contained in its level-1 counterpart. That is, if the level-2 predictor has a level-1 counterpart in the model, its level-2 effect will be a total level-2 between effect if its counterpart contains no level-2 between variance (as created through an observed variable or through model-based partitioning of its variance), but the level-2 predictor will have a level-2 contextual (incremental between) effect otherwise.
  - In a three-level model, you do have to worry about smushing a level-2 predictor's effect. What its effect actually means then depends on two things: (1) how the level-2 variable is modeled (i.e., as an observed predictor or as a level-2 outcome whose variances at levels 2 and 3 are estimated as part of the likelihood), and (2) how its level-2 effect is specified in the syntax.
  - For a level-2 predictor whose variance is NOT partitioned by the model: when specified using either direct method (a) or placeholder method (b), the interpretation of the level-2 predictor's fixed effect is determined by the variance it contains, the same as in other programs (and as described in chapter 11).
    - If the level-2 predictor has been variable-centered (i.e., its level-3 variance has been removed), its level-2 effect will be the "pure" level-2 between effect.
    - If the level-2 predictor has been left uncentered or centered at a constant (i.e., grand-mean-centering in which its level-3 variance is still present), its level-2 effect will be a smushed effect UNLESS any variance it has at level 3 is included through a model predictor its level-3 contextual effect, in which case the effect of the level-2 predictor is then its "pure" level-2 between effect as intended.
  - For a level-2 predictor whose variance at levels 2 and 3 IS partitioned by the model, references to the variable at each level actually refer to its partitioned variance at that level. Consequently, the level-2 predictor's effect is specifically for its level-2 variance as the predictor, such that it carries only the "pure" level-2 between effect. However, the interpretation of its level-3 effect depends on how the level-2 effect was specified:
    - If using direct method (a), the level-3 effect is the total level-3 between effect (i.e., not controlling for the absolute level-2 predictor value).
    - In contrast, when using placeholder method (b), the level-3 effect is the contextual level-3 (incremental between) effect instead (i.e., after controlling for the absolute level-2 predictor value). This is the case regardless of whether its level-3 random effect variance—differentially allowed by use of the method (b) placeholder syntax for the level-2 effect—is estimated in the model.

- **Specifying and interpreting fixed effects of level-3 predictors in three-level models**
  - The following applies to both main effects of level-3 predictors and to interactions among level-3 predictors (the latter must be represented explicitly using new separate variables).
  - A level-3 variable will be used as an observed level-3 predictor if it is included on the **`BETWEEN =`** subcommand of the **`VARIABLE:`** command, in which case it is not included in the likelihood, it has no distributional assumptions, and any cases with missing values will be listwise-deleted. As a second option, a level-3 predictor listed on the **`BETWEEN =`** subcommand whose mean, variance, or covariance is referred to explicitly in the syntax will be brought into the likelihood at level 3 only. This means that its variance will be estimated, so distributional assumptions still apply and its values can be missing at random without listwise deletion. However, how to add effects of level-3 variables in the model proceeds similarly in either of these two cases.
  - In contrast to level-1 or level-2 predictors, the effect of a level-3 predictor can only be added directly using the keyword **`ON`** through direct method (a).

- For example using direct method (a), to add a level-3 fixed effect predicting `y1` from level-2 predictor `L3x`:
  - Under the `%BETWEEN L3IDname%` level-3 model, a fixed effect is added: `y1 ON L3x;`
- In a three-level model, level-3 predictors only contain level-3 variance, and thus can only have level-3 effects (i.e., you do not have to worry about smushing them). However, the effect of a level-3 variable with a counterpart at level 1 and/or level 2 will have a different interpretation depending on the variance contained in its lower-level counterparts. That is, if the level-3 predictor has a level-2 or level-1 counterpart in the model, its level-3 effect will be a total level-3 between effect if its counterparts contain no level-3 between variance (as created through an observed variable or through model-based partitioning of their variance), but the level-3 predictor will have a level-3 contextual (incremental between) effect otherwise.

Adding Parameter Constraints and Requesting Model-Implied New Parameters (within any kind of model, not just multilevel models)

- MPLUS syntax contains a labeling convention by which to constrain model parameters to equality or to use them in creating new model-implied parameters. To do so, a label of 8 or fewer characters is added in parentheses before the semi-colon on the same physical line of syntax in which an estimated parameter is referred.
- For example, to creates labels of `resvar1, resvar2…` `resvar6` for the variances of `var1, var2…` `var6`:
  `var1-var6 (resvar1-resvar6);`
- However, to force all six variances to be equal, a single `resvar` label is created instead:
  `var1-var6 (resvar);`
- For example, to create a label of `int` for the intercept of `y1`, of `bx1` for the slope of `x1` predicting `y1`, and of `bx2` for the slope of `x1` predicting `y1`: `[y1] (int);` `y1 ON x1 x2 (bx1 bx2);`
- However, to constrain the effects of `x1` and `x2` to be equal in predicting `y1`, a single label of `bx` is created for the common slope instead: `y1 ON x1 x2 (bx);`
- These labels can then be used to create new model-implied parameters whose estimates, standard errors, and corresponding Wald test *p*-values will then be provided in the output. The `MODEL CONSTRAINT:` command and `NEW ( )` subcommand (which will each appear in black font) first tell MPLUS that additional model-implied parameters will be requested, and then those new parameters can be defined mathematically.
- For example, to estimate new predicted `y1` values if `x1` = 10 and `x2` = 5, and if `x1` = 5 and `x2` = 10 called `y1if105` and `y1if510`, respectively:

  -
    ```
    MODEL:
    [y1] (int);
    y1 ON x1 x2 (bx1 bx2);

    MODEL CONSTRAINT:
    NEW (y1if105 y1if510);
    y1if105 = int + bx1*10 + bx2*5;
    y1if510 = int + bx1*5 + bx2*10;
    ```
- These commands can also be used to enforce more specific constraints without creating new model-implied parameters. For example, to tell MPLUS to force the effect of `x2` predicting `y1` to be twice as large as the effect of `x1` predicting `y1`:

  -
    ```
    MODEL:
    [y1] (int);
    ```

```
y1 ON x1 x2 (bx1 bx2);

MODEL CONSTRAINT:
bx2 = 2*bx1;
```

Requesting Multivariate Wald Tests (within any kind of model, not just multilevel models)

- Multivariate Wald tests of multiple fixed effects simultaneously can be requested via a `MODEL TEST:` command (which will appear in black font), which makes use of the same practice of labeling parameters as just shown. However, only one `MODEL TEST:` command is allowed per model.
- For example, to request a Wald test with two degrees of freedom for the whether both slopes = 0 for `x1` and `x2` predicting `y1` simultaneously:

  -
    ```
    MODEL:
    [y1] (int);
    y1 ON x1 x2 (bx1 bx2);

    MODEL TEST:
    bx1=0;
    bx2=0;
    ```

Model-Specific Notes by Chapter Example

- **A note about model fit, degrees of freedom, and information criteria across programs:**
  - As presented in chapters 3 and 5, **AIC** is computed as AIC = −2LL + 2*df, in which *df* = degrees of freedom. But within most programs (including SPSS and SAS but excluding STATA), the degrees of freedom used in computing AIC are different when using ML versus REML estimation. In ML, degrees of freedom include ALL model parameters (fixed effects in the model for the means plus all variances, covariances, and other variance model parameters), but will exclude the fixed effects when using REML. Although this may seem arbitrary, it makes sense if you remember that models that differ in fixed effects cannot have their fit compared using −2LL, AIC, or BIC when using REML (and thus the number of fixed effects is irrelevant for those statistics).
  - As also presented in chapters 3 and 5, **BIC** is computed as BIC = −2LL + Log(N)*df, in which *df* = degrees of freedom (which differs between ML and REML for the same programs as noted for AIC), and *N* refers to some kind of sample size. But what value of *N* is used differs by program. In SPSS and MPLUS, *N* = the total number of observations, whereas *N* = the sample size at the highest level of the model in SAS. More specifically, in SAS *N* is the number of unique repetitions of the **V** matrix, which means that *N* = 1 in models with crossed random effects. In STATA, the user can tell the program what *N* should be (and in my examples, I follow the SAS convention). Consequently, BIC will differ across these programs for any multilevel model.
- **Fake people:** Note that the method of "fake people" to create predicted outcomes is not as easy to implement in MPLUS and was thus not included. However, the `PLOT:` command does allow you to create and plot predicted values given specific values of model predictors. `MODEL CONSTRAINT:` could also be used to create each predicted value (with one line of syntax required for each value).

- **Chapter 2**
  - Because chapter 2 includes single-level models only, no `%WITHIN%` and `%BETWEEN%` level-specific models were needed.
  - Interaction terms are all at the same level, and thus were each created using `DEFINE:` to make new predictor variables.
  - No predictors were brought into the likelihood in any model, thus any cases with missing predictors would be listwise-deleted, and predictor variables have no distributional assumptions.

- **Chapter 3a**
  - The `DATA WIDETOLONG:` command was first used transpose the data from multivariate/wide (one row per person) to stacked/long (two rows per person).
  - Level-specific models were then created using the `%WITHIN%` and `%BETWEEN%` level-specific syntax. The single-level models were estimated by constraining any level-2 random effects variances to 0.
  - Note that an intraclass correlation (ICC) is provided descriptively by default in the output given the use of the `%WITHIN%` and `%BETWEEN%` level-specific syntax, not the specific model being estimated. For instance, an ICC is still provided in the output for the between-person models, even though those models predict that ICC = 0.

- **Chapter 3b**
  - Because of how MPLUS formats its output, the response time (RT) outcome, originally scaled in milliseconds, was divided by 10 so that its model parameters could be seen on the output (omitting this step results in fields of asterisks instead because the values are too large to be printed).
  - A non-diagonal **R** matrix is not possible in MPLUS using the multilevel modeling level-specific syntax, so `DATA LONGTOWIDE:` was first used transpose the data from stacked/long (six rows per person) to multivariate/wide (one row per person). RT at the six sessions was then represented by six separate variables and models were specified using single-level `MODEL:` syntax.
  - In the Between-Person Independent ANOVA and Univariate Repeated Measures ANOVA models, the label (`totvar`) was used to constrain the six variances to be equal. In the former model, all covariances were constrained to 0, whereas in the latter model, all covariances were constrained to be equal using the label (`totcov`). In the Multivariate Repeated Measures ANOVA models, there were no constraints—the labels indicated six separate variances and 15 separate covariances.
  - In each model, the six means were estimated separately (labeled as `mean1-mean6`).
  - The `MODEL TEST:` command was used to create a multivariate Wald test of all possible mean differences among the six RT outcomes, the same as is reported in a typical ANOVA model.

- **Chapter 4**
  - A non-diagonal **R** matrix is not possible in MPLUS using the multilevel modeling level-specific syntax, so the `DATA LONGTOWIDE:` command was first used transpose the data from stacked/long (seven rows per person) to multivariate/wide (one row per person). Positive mood (`posmood`) for the seven days was then represented by seven separate variables and models were specified using single-level `MODEL:` syntax.
  - For each model, the number of distinct labels used indicated the number of parameters estimated and any constraints thereof.

  - In the Unstructured **R** matrix models, the label (`totvar1-totvar7`) referred to the seven separate variances and the label (`totcov1-totcov21`) refers to the 21 separate covariances. In contrast, in the homogeneous variance **R** matrix models, the label (`totvar`) referred to the single constrained variance estimate shared across the seven days.
  - Similarly, in the saturated means model, the label (`mean1-mean7`) referred to the seven separate means, whereas in the empty means models, the label (`mean`) referred to the single constrained mean shared across the seven days.
- **R** matrix covariance patterns were created using a combination of distinct labels whose values were then given within the `MODEL CONSTRAINT:` command as a function of existing labeled parameters and new parameters:
  - In the CSH **R** matrix model, the covariance between each pair of days was formed from a common compound symmetry correlation (a new parameter called `CScor`) and the labeled estimated variance at those occasions.
  - In the AR1 **R** matrix models, the covariance between each pair of days was formed from a common auto-regressive correlation (a new parameter called `AR1cor`) and the labeled estimated variance at those occasions (either constant across days in the AR1 homogeneous variance model, or heterogeneous across days in the ARH1 heterogeneous variance model).
  - In the Toeplitz **R** matrix models, the covariance between each pair of days was formed from common Toeplitz lag-specific correlations (a new parameter called `TOEPcorX` for each lag X) and the labeled estimated variance at those occasions (either constant across days in the homogeneous variance models, or heterogeneous across days in the heterogeneous variances models).
- In the **G** and **R** combined models, a different convention based on structural equation modeling syntax was used to create a random intercept variance within the **G** matrix than was used in the level-specific multilevel modeling syntax used in later chapters:
  - Specifically, a latent factor for the random intercept was created using the `BY` statement, in which the factor loadings for the seven outcomes were each constrained to 1: `RandInt BY posmood1-posmood7@1;`
  - Consequently, after being predicted by the random intercept latent factor, the variance that remains in the seven observed variables for positive mood is their residual variance for the **R** matrix specifically. Thus, the constraints used to create covariance patterns among the observed variables in these **G** and **R** combined models do so specifically for the residuals—as denoted in the labels by using `res` to indicate *residual* variance and covariance instead of `tot` to indicate *total* variance and covariance.
  - Note that the highest-lag residual covariance was constrained to 0 in each of the **G** and **R** combined models.
  - In order to estimate a fixed intercept for the common mean across days via the mean for the random intercept latent factor, the intercepts for the observed variables were constrained to 0. This is really just a semantic distinction, but it maps more directly onto the idea of fixed intercept as a mean and random intercept variance as the latent factor variance that is used in the multilevel modeling syntax.

- **Chapter 5**
  - Multilevel modeling syntax was used for the fixed and random linear time models with a diagonal **R** matrix, in which predicted means at each occasion were estimated in the `MODEL CONSTRAINT:` command using the labeled fixed intercept (`int`) and fixed linear time slope (`btime`). In general, fixed effects were labeled to begin with `b` followed by the predictor they referred to.
  - For the saturated means, unstructured **R** matrix model, single-level model syntax again was used on restructured data (transposed from stacked to multivariate using the `DATA LONGTOWIDE:` command).

- For the last two models including a non-diagonal **R** matrix, the same multivariate data structure was again used. Consequently, the random effects were specified using structural equation modeling syntax (as was used in the **G** and **R** combined models in chapter 4):
  - A random intercept latent factor was created using the `BY` statement, in which the factor loadings for the four occasions as separate outcome variables were each constrained to 1:
    ```
    RandInt BY outcome1-outcome4@1;
    ```
  - A random linear time slope latent factor was also created using the `BY` statement, in which the factor loadings for the four outcome variables were each constrained to match their time values:
    ```
    RandLin BY outcome1@0 outcome2@1 outcome3@2 outcome4@3;
    ```
  - Consequently, after being predicted by the random intercept and random time slope latent factors, the variance that remains in the four outcome variables is their residual variance for the **R** matrix specifically.
  - The AR1 and TOEP2 residual covariance patterns were then created through the use of labels and the `MODEL CONSTRAINT:` command as in chapter 4.

- **Chapter 6**
  - Because of how MPLUS formats its output, the response time (RT) outcome, originally scaled in milliseconds, was divided by 10 so that its model parameters could be seen on the output (omitting this step results in fields of asterisks instead because the values are too large to be printed).
  - Multilevel modeling syntax was then used for all polynomial and piecewise random effects models. The specific time predictors needed (e.g., linear and quadratic time, piecewise time slopes) were created using the `DEFINE:` command, and predicted intercepts, slopes, or differences between slopes were created via the `MODEL CONSTRAINT:` command using the labeled fixed effects.
  - In order to create the nonlinear constraints necessary for the negative exponential models, the data were first restructured from stacked to multivariate using the `DATA LONGTOWIDE:` command so that session-specific means, variances, and covariances could be labeled; how the model should predict them was then specified through `MODEL CONSTRAINT:`. Unfortunately, only the random asymptote model would converge.

- **Chapter 7a**
  - Saturated means models were again estimated on multivariate data transposed using `DATA LONGTOWIDE:`
  - Multilevel modeling syntax was again used for models examining time-invariant predictors, although the heterogeneous variance versions were not able to be estimated.

- **Chapter 7b**
  - Saturated means models were again estimated on multivariate data transposed using `DATA LONGTOWIDE:`
  - Multilevel modeling syntax was again used for the unconditional models of change and for the models examining effects of time-invariant predictors. The labels for their effects are necessarily becoming more complicated in order to keep track of what each means. For example, the labels for the effects of attitudes have three parts: `b` to indicate a fixed effect coefficient, `att` to indicate the predictor, and then the last part indicates what level-1 term is being predicted (`int` for the intercept, `lin` for the linear slope, or `quad` for the quadratic slope).

- **Chapter 8**
    - Multilevel modeling syntax was used for models examining time-varying predictors, although the heterogeneous variance versions were not able to be estimated.
    - All level-1 predictors were treated as observed variables that were not included in the likelihood.
    - Interactions among variables at the same level were created as separate observed variables, whereas cross-level interactions were specified within the syntax (and thus level-1 effects were specified using the placeholder method (b) that allows cross-level interactions with them). The same will be true in later chapters.

- **Chapter 9**
    - Multilevel modeling syntax was used for the univariate models examining time-varying predictors.
    - In addition, the truly multivariate longitudinal models were estimated using multilevel modeling syntax. In these models, both time-varying variables (the risky behavior outcome and the monitoring predictor) were included in the likelihood, such that their random intercept, random linear age slope, and residual variances were partitioned by the model. Note that the directed effects version (i.e., in which monitoring predicts risky behavior) were not able to be estimated in the other programs, although the undirected effects version (in which the relationships between monitoring and risky behavior were specified using covariances instead) was estimated in SAS and SPSS by "tricking" a univariate model into becoming a multivariate model, as was described in chapter 9.
    - For demonstrative purposes, the same truly multivariate longitudinal models were estimated using structural equation modeling syntax as well (including both the direct effects and undirected effects versions).
    - The slopes-as-outcomes models were not estimated given the extra data manipulation steps they would have required to save the estimates and read them back into MPLUS as data. Given that these models cannot be recommended in the first place, I didn't bother.

- **Chapter 10a**
    - Multilevel modeling syntax was used for models examining accelerated time, with no new twists.

- **Chapter 10b**
    - Three-level versions of the multilevel modeling syntax were needed for chapter 10b.
    - Note that an intraclass correlation (ICC) for each higher level is provided descriptively by default in the output given the use of the **`%WITHIN%`** and **`%BETWEEN%`** level-specific syntax, not the specific model being estimated. For instance, an ICC for level 3 is still provided in the output for the two-level models, even though those models predict that the level-3 ICC = 0. However, the ICCs provided for three-level models in MPLUS are actually just the proportion of variance at that level out of the total variance. So the level-2 ICC is not the expected correlation of any kind, although the level-3 ICC (denoted ICC_{L3b} in chapter 11) is the correlation of level-1 units from the same level-3 unit (but not from the same level-2 unit).
    - Given that the saturated means models included only random intercepts at levels 2 and 3 in the **G** matrix along with a diagonal **R** matrix, they were estimated within the stacked data by creating dummy-coded predictors to differentiate sessions and bursts in which the last value served as the reference for each dimension of time.
    - After estimating a separate series of univariate models for symptoms and for positive affect, a three-level truly multivariate model was estimated in which both variables were treated as level-1 outcomes in the likelihood and all necessary fixed and random effects related to sessions and bursts were included simultaneously. Relationships between the two variables were modeled via covariances in the **G** and **R** matrices rather than directed effects.

- **Chapter 11a**
  - Three-level versions of the multilevel modeling syntax were again needed for chapter 11a.
  - Given that the saturated means models included **G** and **R** matrices for classes and students unstructured across occasions, they were estimated using two-level model syntax on a separate dataset with a multivariate structure for occasions, but a multilevel structure for students and classes (i.e., one student per row, but occasions as separate outcome variables). This was necessary because the **`DATA LONGTOWIDE:`** command is not available when estimating multilevel models.
  - All level-1 and level-2 predictors were treated as observed variables that were not included in the likelihood.
  - Models using the occasions within classes condensed data structure were not estimated (but one could do so after creating the necessary dataset using two-level syntax).
  - Placeholder method (b) notation was used for lower-level effects when necessary for including their higher-level random effects or cross-level interactions, but direct method (a) was used for lower-level effects when possible otherwise to facilitate ease of estimation.

- **Chapter 11b and Chapter 12**
  - As of MPLUS v. 7.3, models with crossed random effects are only able to be estimated using **`the ESTIMATOR = BAYES`** subcommand instead of **`ESTIMATOR = ML`** for maximum likelihood. Given that this text did not address the extra steps needed to conduct a Bayesian analysis, syntax for these models is not provided.