# Introduction to Latent Transition Analysis

## Dr Oliver Perra: o.perra@qub.ac.uk

We are going to use Mplus to conduct some analyses in a dataset. The dataset is provided with the exercises and it is called:

**NLS\_extract.dta.dat**

The dataset includes an extract of some longitudinal data collected for NLS in 1986, 1988, and 1990. The data I will focus on concern reports of behaviour and emotional difficulties (e.g. Anxiety and Depression).

In what follows, I will provide an introduction to Mplus and its syntax. If you are familiar with Mplus, you can skip this.

I will then provide an introduction concerning the syntax used to specify latent class and latent transition models in Mplus.

# Introduction to Mplus - For Mplus beginners

Mplus operates by creating INPUT files: these instruct the software on how to read the file, the type of variables, the type of analysis, algorithms and estimators to be used, the model to be estimated. The input file also instructs the software on the results and parameters to be displayed in the OUTPUT file, how to create files with graphics, how to create other files with derived parameters from the models estimated.

Once the analyses have been run, Mplus will produce an OUTPUT file. This is text file that reports the instructions provided in the INPUT and the results of the analyses run.

The principal commands in Mplus input files are:

TITLE:

DATA:

VARIABLE:

ANALYSIS:

MODEL:

OUTPUT:

PLOT:

SAVEDATA:

These are all followed by a colon.

Within each command, it is possible to provide further options. These are separated by semicolon “;” at the end of each option.

Other common rules in Mplus are that the verbs “are/is” and the equal sign “=” are allowed to specify some options. For example, in specifying the file where the data is one can write :

DATA: File is dat.dat;

or

DATA: File = dat.dat;

Hyphens (-) are used to indicate a list of variables or numbers. Items in a list can be separated by space or commas.

Exclamation marks (!) are used to write comments in the input that are not read by the software. What follows the “!” in the line is skipped by the software.

**A SHORT INTRODUCTION TO MPLUS SYNTAX**

Some general rules:

* The main commands are followed by colon “:” . Options within the commands are followed by semi-colon “;”.
* Most Mplus options can be shortened (e.g.: USEVARIABLE=🡪USEVAR=).
* It is possible to use lower- or upper-case characters to name variables, but in the outputs Mplus will report the variable names using upper-case characters.
* It is possible to give names to variables that exceed 8 characters, but in the outputs Mplus will only report the first 8 characters of a variable name.

**TITLE:**

This command allows you to provide a title to the file (free text).

**DATA:**

This command is used to specify where the data are (i.e. the data file to be read), their format and other options (if necessary). It is common to create .dat files (or .txt) that contain data in person-level format (one row for each individual, with variables values in columns). Do not include variable names in the data file !!!

It is possible to specify a path to the data file (e.g.: C:\Desktop\datac1.dat).

NOTE: If there are spaces in the path to the data file, you must enclose the path between parentheses, e.g.:

DATA: FILE=”C:\my files\datac1.dat”

An important option in the DATA: command is **LISTWISE= ON**, which instructs Mplus to conduct analyses with listwise deletion of cases with one or more missing data in the variables of interest. From version 5 of Mplus the default option is to include cases with incomplete information, i.e. with some missing values on some of the variables in the model.

**VARIABLE:**

This command is used to provide names to the variables in the dataset and specify what types of variables they are (their scale) etc.

Important options are:

**NAMES =** 🡪 list the names of the variables in the dataset.

e.g.:

NAMES = cpubid\_xrnd mpubid\_xrnd crace\_xrnd csex\_xrnd cyrb\_xrnd cogna86 emota86 ;

**USEVARIABLE =** 🡪 lists the variables that will be used in the analyses or models. For example, if you will run analyses on the items concerning respondents’ strengths and difficulties, write:

USEVARIABLES = ranti86 ranx86 rhypr86 rdep86 rpeer86;

This will ensure other variables, e.g. IDs, will not included in the analyses.

Note that if a variable is specified after USEVARIABLE= it will be included in the models specified after command **MODEL:** even if not mentioned specifically in the **MODEL:** command.

**CATEGORICAL, NOMINAL, COUNT =** 🡪 are used to define a list of *dependent* variables as ordered categorical (or binary), nominal (unordered), or count variables respectively. Categorical, nominal and count variables that are not treated as dependent should NOT be listed under the CATEGORICAL etc. command.

e.g.:

CATEGORICAL = ranti86 ranx86 rhypr86 rdep86 rpeer86;

**MISSING=** 🡪 specifies the value (or a character such as . or \*) used to identify a missing value for one or more variables. If all the variables use the same missing value indicator (e.g. -999), write:

MISSING = all (-999);

**IDVARIABLE=** is used to indicate the identifier for each observation or case in the dataset. This is necessary if you want to create data files after the analyses (**SAVEDATA:** command) and want to use them for further analyses: in this case, the file saved with **SAVEDATA:**  will contain IDs for the observations. e.g.:

IDVARIABLE = cpubid\_xrnd;

**CLUSTER=** is used to indicate a clustering variable (e.g. school, group). This is necessary for multilevel analyses or for analyses that adjust for clustering (option: **COMPLEX** in **ANALYSIS:** command). For example, you could specify that data are clustered at the county level:

CLUSTER=county;

*!remember that to adjust for clustering, you need to also add the option COMPLEX after ANALYSIS:*

**CLASSES** is used to specify names of latent categorical variables and the number of categories (between parentheses). If we want to estimate a model with a latent variable called “class” with 2 categories we could write:

CLASSES = class (2);

We can add more latent variables after **CLASSES=**, depending on our model. For example, if we wanted to specify a latent class model with latent classes for data collections in the year 1986 and in the year 1988, we could write:

CLASSES = class86 (2) class88(3);

The text above instructs Mplus to estimate a latent class variable with two classes called “class86” and a second latent class variable that includes three classes, “class88”. To specify the associations between these variables and the associations between them and the observed variables (indicators), we will have to use the **MODEL:** command (see below and the section “Introduction to Latent Class syntax in Mplus”).

**ANALYSIS:**

This is the command used to specify the type of analysis and other options in the analysis (e.g. type of estimator).

Option **TYPE=** invokes a specific type of analysis among a range of options (e.g. EFA, i.e. exploratory factor analysis).

**TYPE=BASIC;**

invokes *descriptive statistics* on the variables included by USEVARIABLE.

In order to run Latent Class Analysis, Latent Transition Analysis, or other mixture models (GMM or LCGA, etc.) one has to invoke **TYPE=MIXTURE;**

**MODEL:**

The model command allows to specify a model, constrain parameters, test parameter constraints.

Mplus allows to specify many parameters of a model using syntax. The main operators in the syntax are:

**ON:** short for “regressed on” and defines regression relationships e.g.:

MODEL:

y ON x male;

will estimate the regression of variable “y” on variables “x” and “male”.

**BY:** is short for “measured by”, and defines latent continuous variables (but not latent classes!). For example:

MODEL:

f1 BY y1 y2 y3 y4 y5;

will estimate a latent variable “f1” using indicators “y1”, “y2”, etc.

Note that the same command can be shorted in this way:

f1 BY y1-y5;

where the hyphen indicates all variables from y1 to y5.

**WITH:** indicates correlational relationships. For example:

MODEL:

f1 WITH f2;

indicates a correlation between f1 and f2.

In the **MODEL:** command, the name of variables is used to indicate the variance or residual variance (depending on the model) of variables, while the name of a variable between square brackets refers to means, intercepts, or thresholds.

For example, if “x” and “y” are continuous variables, the command:

MODEL:

[x]; x; [y]; y;

will estimate the mean and variance of x and the mean and variance y, respectively.

However, if we added this line:

MODEL:

**y ON x;**

[x]; x; [y]; y;

the last line will estimate the mean and variance of “x”, and the intercept and residual variance of “y”.

The notation above can be used to constrain parameters using other characters.

The character “at” **@** followed by a number is used to constrain a parameter to a specific value. For example:

MODEL:

y ON x;

[x]; x; [y]; **y@0;**

constrains the residual variance of “y” to be equal to 0.

Adding numbers between parentheses after a parameter can be used to impose equality constraints. For example:

MODEL:

y ON x (1);

w ON z (1)**;**

constrains the linear regression of “y” on “x” to be equal to that of “w” on “z”.

The same equality constrains can be applied by using letters or combination of letters and numbers between parentheses. For example, the equality constraint imposed above could be expressed like this:

MODEL:

y ON x (eq1);

w ON z (eq1)**;**

As long as the string of characters and/or numbers between parentheses is the same, equality constraints are imposed.

**OUTPUT:**

Specifies the information to be reported in the output file.

**PLOT:**

Specifies information necessary to allow MPlus to create plots and graphs.

In Latent Class Analysis, the most useful type of plots are **PLOT3**, which include estimated probabilities for a categorical latent variable as a function of its covariates, and latent variable distribution plots.

When using PLOT3 is also advantageous to list the names of the set of variables that can be used in plots where their values are connected by a line, e.g. the indicators of the latent class model.

To do so use the option SERIES. You can also specify where the values of this variable can be in the x axis of a graph by following the name of the variable with the desired x-axis value between parenthesis.

For example, the following command will ensure you can create PLOT3 types, with the first variable in the list appearing at point 1 of the x-axis, the second appearing at point 2 of the x-axis, etc.:

PLOT: Type = PLOT3;

SERIES= ranx86(1) rdep86(2) ranti86(3) rhypr86(4) rpeer86(5);

If you wanted to represent these variables at 0.5 units of the x-axis, starting from 1, you can write:

PLOT: Type = PLOT3;

SERIES= ranx86(1) rdep86(1.5) ranti86(2) rhypr86(2.5) rpeer86(3);

**SAVEDATA:**

Instructs the programme to create a file with parameters estimated.

**FILE =** used to give a name to new file. For example:

FILE IS analsis1.dat;

**SAVE =** uses to specify what information can be saved in the file. **CPROB** will save posterior latent class probabilities and modal class assignment for each individual included in the analyses. If an IDVARIABLE is specified in **VARIABLE:** command, the file will contain ID variable information, and it will be possible to match-merge the file with other files for further analyses. E.g.:

FILE is analysis1.dat;

SAVE = CPROB;

# Introduction to latent class syntax in Mplus

## **Estimating latent class variables**

Mplus estimates latent class models using a combination of the **VARIABLE:** and **ANALYSIS:** commands.

Namely, when you want Mplus to estimate a latent variable model, in the **VARIABLE:** command you should include a line that defines the name of the latent class variables and the number of classes that must be estimated.

If you have a similar syntax to define the variables in the dataset

VARIABLES:

NAMES = […] ;

MISSING= all (-999);

USEVAR = ranti86 ranx86 rhypr86 rdep86 rpeer86;

CATEGORICAL=ranti86 ranx86 rhypr86 rdep86 rpeer86;

and you wanted to estimate a latent class model with two categories, you will add the last line as below:

VARIABLES:

NAMES = […] ;

MISSING= all (-999);

USEVAR = ranti86 ranx86 rhypr86 rdep86 rpeer86;

CATEGORICAL=ranti86 ranx86 rhypr86 rdep86 rpeer86;

**CLASSES=c(2);**

The last line will estimate a latent variable “**c**” with 2 categories (2 classes). The latent class variable can be called whatever you prefer (considering Mplus variables are restricted to 8 characters in the Output file) and you can use the syntax to specify different number of classes. For example, if I wanted to call the latent variable “**disorder**” and estimate 6 classes, I would write:

VARIABLES:

NAMES = […] ;

MISSING= all (-999);

USEVAR = ranti86 ranx86 rhypr86 rdep86 rpeer86;

CATEGORICAL=ranti86 ranx86 rhypr86 rdep86 rpeer86;

**CLASSES=disorder(6);**

In ***Latent Transition Analysis*** we want to estimate more than one latent class variable. For this purpose, we need to provide the names and their respective number of categories of at least two latent class variables.

For this example, refer to the data in:

**NLS\_extract.dta.dat**

and the relative document that details the meaning of the variables.

If I wanted to estimate a latent class model of participants’ difficulties when surveyed in 1986, and a latent class model of participants’ difficulties when surveyed in 1988, I must firstly ensure that all these indicators are listed as “USERVARIABLES=”:

VARIABLES:

NAMES = […] ;

MISSING= all (-999);

**USEVAR=ranti86 ranx86 rhypr86 rdep86 rpeer86**

**ranti88 ranx88 rhypr88 rdep88 rpeer88;**

CATEGORICAL=ranti86 ranx86 rhypr86 rdep86 rpeer86

ranti88 ranx88 rhypr88 rdep88 rpeer88;

The commands above instruct Mplus to consider as dependent categorical variables the report of antisocial behaviour collected in 1986 (“**ranti86**”) and in 1988 (“**ranti88**”), and so on.

To specify that I want estimate two latent class variables, and that these will have 2 and 3 classes respectively, I will add the following line:

VARIABLES:

NAMES = […] ;

MISSING= all (-999);

USEVAR=ranti86 ranx86 rhypr86 rdep86 rpeer86

ranti88 ranx88 rhypr88 rdep88 rpeer88;

CATEGORICAL=ranti86 ranx86 rhypr86 rdep86 rpeer86

ranti88 ranx88 rhypr88 rdep88 rpeer88;

**CLASSES= x(2) y(3);**

The last line instructs Mplus to create latent class variable called “**x**” with 2 classes, and latent class variable called “**y**”, which has 3 classes.

You can call these latent variables any name, for example, they could have been called “**disor86**” and “**disor88**”, or any other string of letters and numbers:

**CLASSES= disor86(2) disor88(3);**

Specifying the name and number of latent classes in the **VARIABLE:** command is not sufficient for Mplus to estimate a latent class model. The other necessary step is specifying a mixture model in the option ANALYSIS:, like this:

VARIABLES:

NAMES = […] ;

MISSING= all (-999);

USEVAR=ranti86 ranx86 rhypr86 rdep86 rpeer86

ranti88 ranx88 rhypr88 rdep88 rpeer88;

CATEGORICAL=ranti86 ranx86 rhypr86 rdep86 rpeer86

ranti88 ranx88 rhypr88 rdep88 rpeer88;

**CLASSES= x(2) y(3);**

**ANALYSIS:**

**TYPE=mixture;**

The option “Mixture” invokes estimation of mixture models like latent class analysis, latent transition analysis, etc. This option can be combined with other options, e.g. TWOLEVEL or COMPLEX.

For example, if the data are clustered (e.g. adolescents clustered across schools), you can control for this clustering by specifying the clustering variable in the **VARIABLE:** command, and including the option **TYPE**=**COMPLEX** in the **ANALYSIS:** command.

The default estimator for MIXTURE models is MLR (maximum likelihood with robust standard errors and chi-square). However, one can invoke another estimator by writing the name of the estimator after **ESTIMATOR =**....;

The default algorithm with MLR is the Expectation-Maximisation (EM) algorithm (but other algorithms can be invoked).

It is possible (and indeed, advised) to change the number of initial stage starts and final stage optimizations of the EM algorithm by using option **STARTS=** in the **ANALYSIS:** command, for example:

STARTS = 100 20;

It is also possible to change the number of initial stage iterations in the EM algorithm using option STITERATIONS, for example specifying:

STITERATIONS = 20;

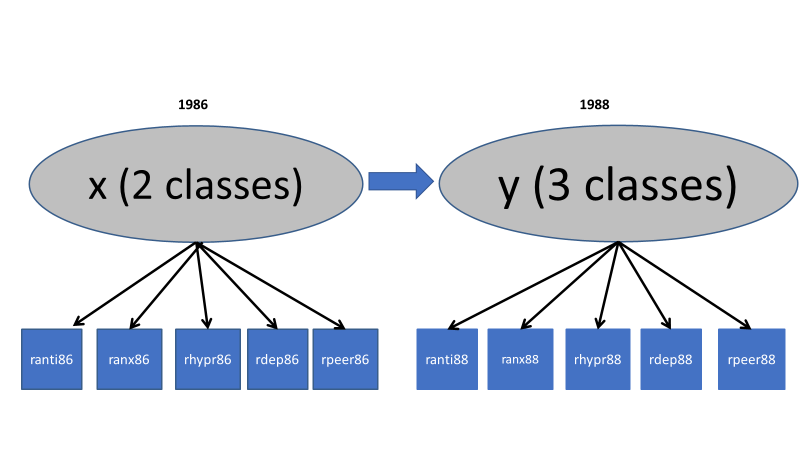
**PROCESSORS**= can also be used to devote more computer memory resources (processors) to the estimation process (default is PROCESSORS = 1).

## **Specify LTA Parameters**

If you are conducting exploratory latent class analysis (i.e. you want to test models with different number of classes) for data collected at one single time point, there is no need to define latent class parameters in the **MODEL:** command.

However, when we are conducting Latent Transition Analyses, we are specifying two or more latent classes: as well as specifying the names of these latent class variables and the number of classes for each of them (as I did above in the **VARIABLE:** command and the **CLASSES=**… ; option), I also must specify that the indicators collected in 1986 relate to one latent class, and the indicators collected in 1988 relate to another latent class.

Namely, the model I want to specify will be similar to this:



*Figure 1: Schematic representation of a Latent Transition model with two data collection points and latent class variables with 2 and 3 classes respectively.*

To specify this model, I will have to use the **MODEL:** command and specify the associations between latent classes and respective indicators at different time points, as well as the associations between the latent classes (if desired).

When we have more than one latent variable, the MODEL: command allows to specify the parameters that must be estimated that *apply to all the latent classes*, as well as the parameters that *pertain exclusively to a specific latent variable*.

To do so, Mplus uses the following options in the **MODEL:** command:

**MODEL:**

**%OVERALL%**

**…**

**MODEL x:**

**….**

**MODEL y:**

**….**

## **Specify Structural Parameters**

Most of the structural parameters can be described in the **%OVERALL%** option of the **MODEL:** command. The information in this portion of the **MODEL:** command includes parameters that concern all the latent variables in the model. This is the point where I can specify that I want to regress my latent variable “**y**” on the latent variable “**x**”. Furthermore, in **%OVERALL%** I can invoke regressions of latent variables on predictors. To specify the structural relations between the latent variables described Figure 1, I will write this:

VARIABLES:

NAMES = […] ;

MISSING= all (-999);

USEVAR=ranti86 ranx86 rhypr86 rdep86 rpeer86

ranti88 ranx88 rhypr88 rdep88 rpeer88;

CATEGORICAL=ranti86 ranx86 rhypr86 rdep86 rpeer86

ranti88 ranx88 rhypr88 rdep88 rpeer88;

CLASSES= x(2) y(3);

ANALYSIS:

TYPE=mixture;

**MODEL:**

**%OVERALL%**

**y on x;**

whereby “**y on x**” asks to estimate the regression of the latent class variable **y** on the latent class variable **x**, the variables I named in the VARIABLE: command.

Since latent variables “**x**” and “**y**” are nominal latent class variables, Mplus will automatically estimate a *multinomial logistic regression* of the 3 latent classes of “**y**” on the 2 latent classes of “**x**”. Mplus considers the last nominal category of a variable as the reference category, therefore the parameters that Mplus will report are the log-odds of being in class 1 of **y** rather than class 3 of **y** (reference category for **y**) if someone is in class 1 of **x** rather than class 2 of **x** (reference category for **x**), and the log-odds of being in class 2 of **y** rather than class 3 of **y** (reference category for **y**) if someone is in class 1 of **x** rathe than class 2 of **x** (reference category for **x**). If I wanted to be more specific and state these parameters in more details in the MODEL: command, I would write the instructions in this way:

VARIABLES:

NAMES = […] ;

MISSING= all (-999);

USEVAR=ranti86 ranx86 rhypr86 rdep86 rpeer86

ranti88 ranx88 rhypr88 rdep88 rpeer88;

CATEGORICAL=ranti86 ranx86 rhypr86 rdep86 rpeer86

ranti88 ranx88 rhypr88 rdep88 rpeer88;

CLASSES= x(2) y(3);

ANALYSIS:

TYPE=mixture;

**MODEL:**

**%OVERALL%**

**y#1 ON x#1;**

**y#2 ON x#1;**

whereby “**y#1 ON x#1**” is invoking the log-odds of being in latent class 1 of **y** rather than latent class 3 of **y** (the reference category of **y**) if someone is in latent class 1 of **x** rather than latent class 2 of **x** (the reference category of **x**), and so on. This more detailed specification of the parameters in the model can be useful if we wanted to impose constraints on them.

## **Specify Measurement Parameters**

The model depicted in Figure 1 assumes that latent class variable “**x**” affects responses in the indicators collected in 1986, and the latent class variable “**y**” affects responses in the indicators collected in 1988. To do so, we should specify measurement parameters for those two models in the sections of the **MODEL:** command that pertain specifically to each latent class, which I have highlighted here:

VARIABLES:

NAMES = […] ;

MISSING= all (-999);

USEVAR=ranti86 ranx86 rhypr86 rdep86 rpeer86

ranti88 ranx88 rhypr88 rdep88 rpeer88;

CATEGORICAL=ranti86 ranx86 rhypr86 rdep86 rpeer86

ranti88 ranx88 rhypr88 rdep88 rpeer88;

CLASSES= x(2) y(3);

ANALYSIS:

TYPE=mixture;

**MODEL:**

%OVERALL%

y#1 ON x#1;

y#2 ON x#1;

**MODEL x:**

**%x#1%**

…

**%x#2%**

**MODEL y:**

**%y#1%**

….

**%y#2%**

….

**%y#3%**

…

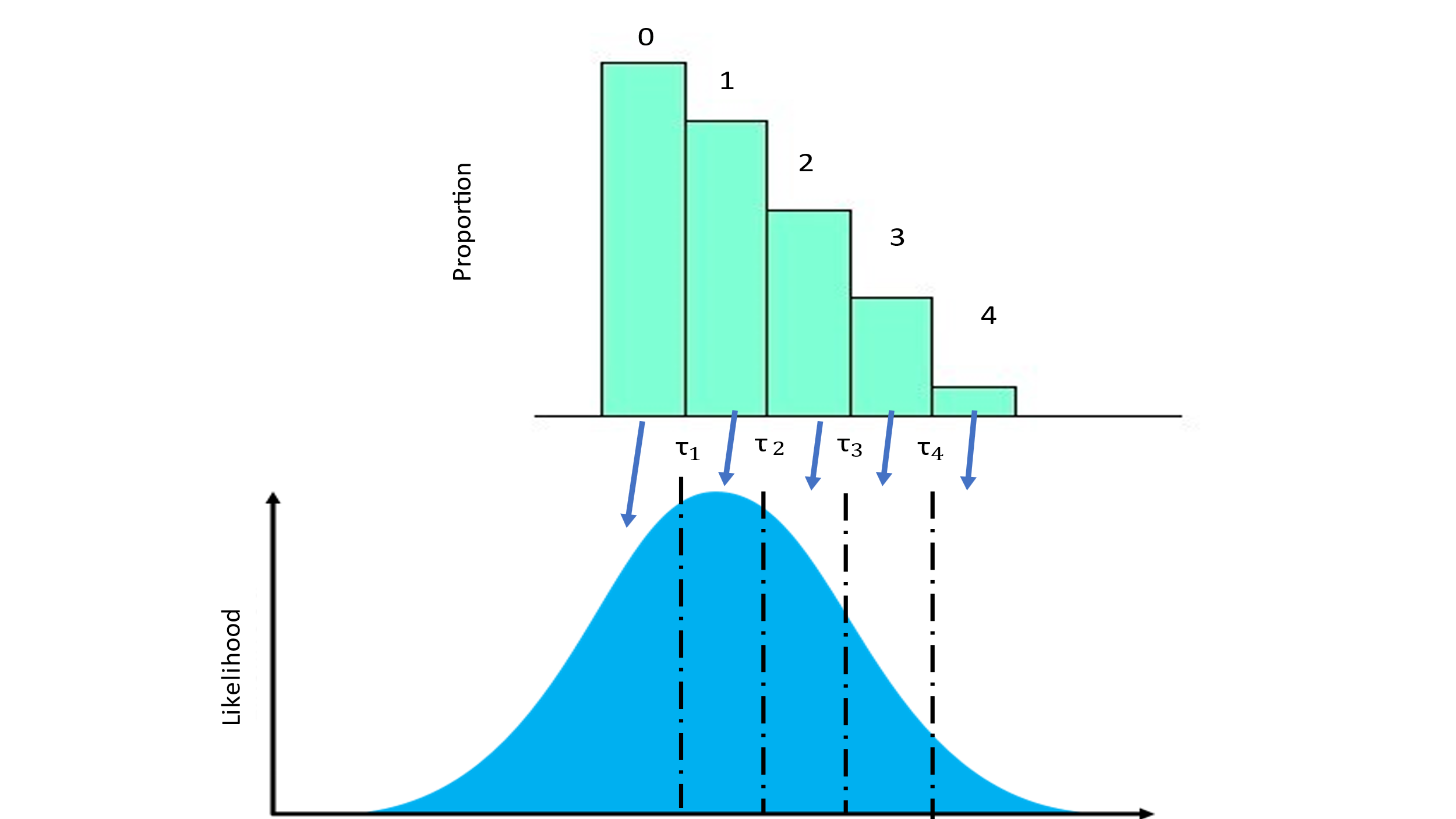
The **MODEL x:** section introduces the parameters that pertain specifically latent variable **x**. Since latent variable **x** is a latent class variable with two latent classes, Mplus expects you to specify the parameters that pertain latent class 1 of **x** after the **%x#1%** line, and the parameters that pertain specifically to latent class 2 of **x** after the **%x#2%** line. Similarly, Mplus expects to be fed parameters for latent class **y** following the **MODEL y:** line, and specifically the parameters for latent class 1 of **y** will feature after the **%y#1%** line, etc.

In the example presented here, I am dealing with categorical indicators, and this requires some explanation on how categorical indicators are linked to latent variables. However, LCA and LTA can be applied to all type of indicators.

## **Specify Measurement Parameters for Categorical Indicators**

When we are considering ordered categorical variables as dependent variables, these are linked to known probability distributions through a *logit link*. In practical terms, observed categorical variables are considered coarse ways to cut an underlying continuous distribution. The cut-points that determine response categories proportions are thresholds *τ,* and these are expressedas logits (log-odds)

To clarify, consider Figure 2:



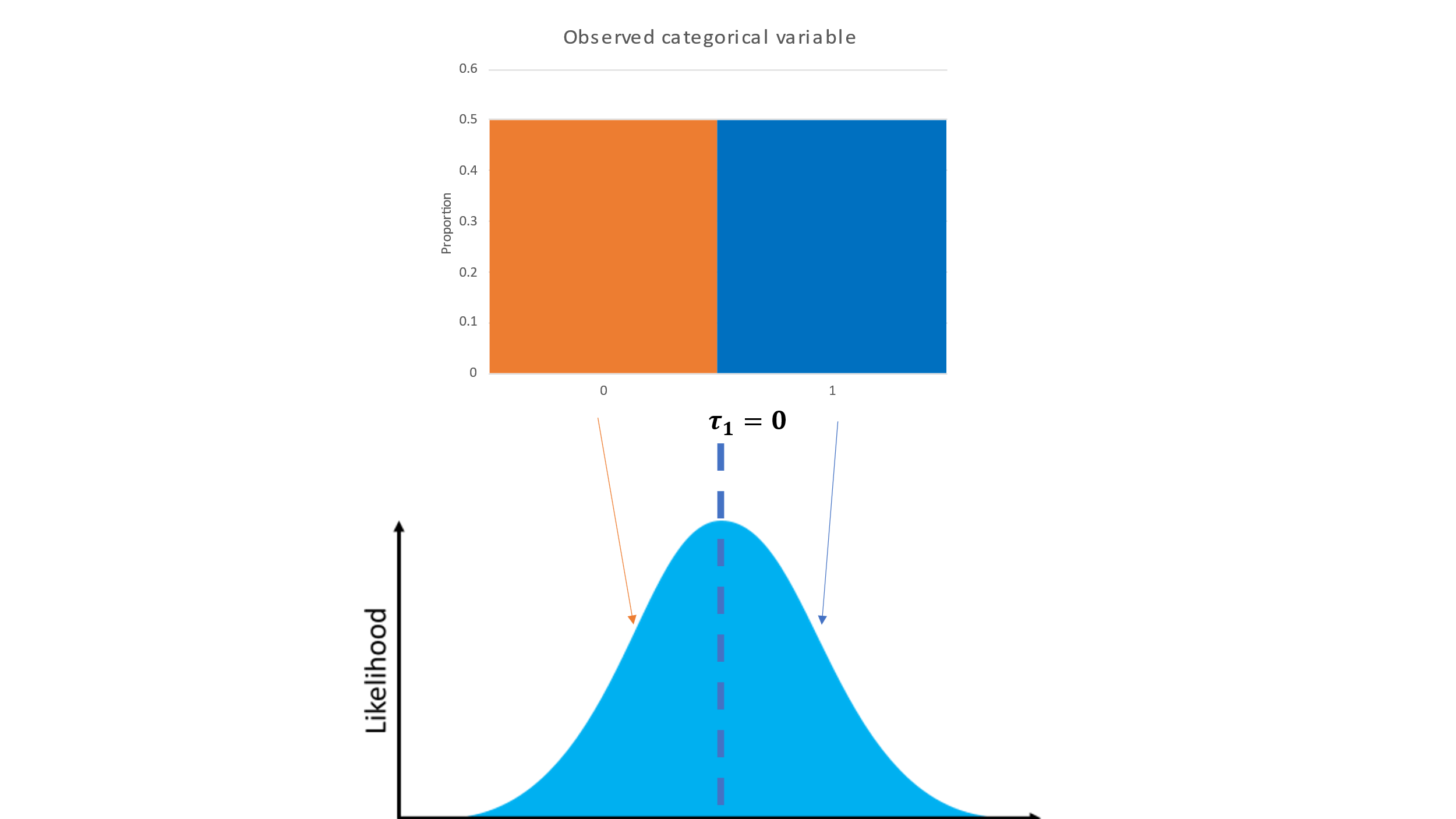
*Figure 2: Pictorial representation of how thresholds () in an underlying continuous latent response variable correspond to the categories in a ordered categorical observed variable.*

Figure 2 illustrates that the categories of the observed categorical variable correspond to different segments of the underlying continuous distribution: The proportions of these categories correspond to the density function within the cut-points of the underlying continuous distribution. The continuous distribution is “cut” into different categories by the thresholds *().* Since there are 5 categories in the example (0;1;2;3;4), 4 thresholds are needed to “cut” the underlying continuous distribution into 5. The frequency of the observed categories thus depends on where we cut the underlying distribution, i.e. where we put the thresholds *()*. For example, had we placed the 4th threshold *()* at a higher level on the x-axis, we would have fewer or no cases into category 4 of the observed categorical variable.

Thresholds are expressed using the logit of a probability, whereby:

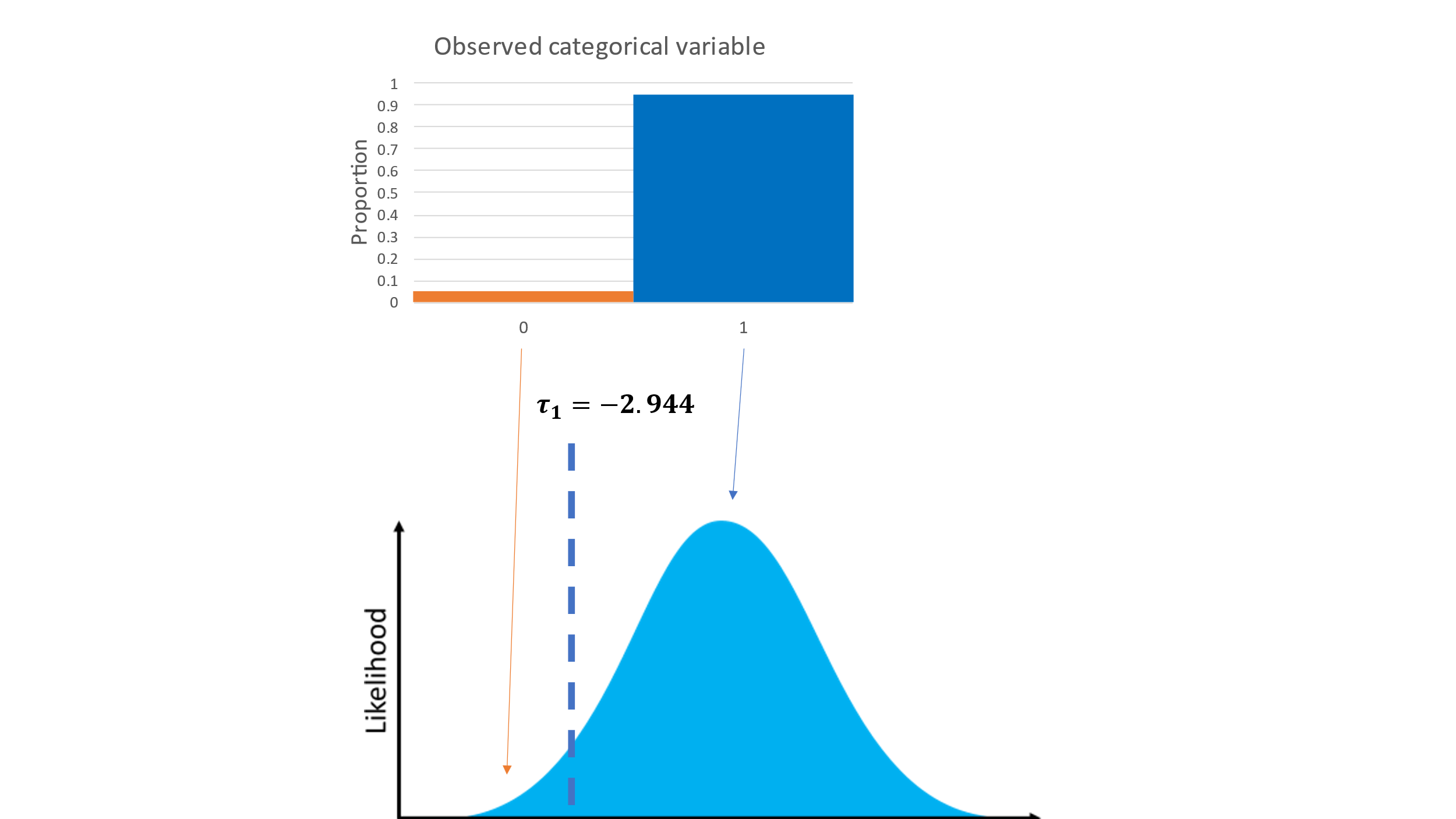
logit(p) = - log ( ;

Logits range from – to . Using the formula above, a probability = .50 corresponds to *logit*=0. For example, if we had a categorical dichotomous variable, only one threshold is necessary for “cutting” the underlying distribution into two categories, and if the threshold is equal to 0, the two categories of the variable have a similar probability, as indicated in Figure 3.



*Figure 3: Pictorial representation of a thresholds () that divides an underlying continuous latent response variable into two categories with the same proportion.*

If the same threshold was equal = -2.95, this will correspond to a probability of the first category in the observed variable equal to 0.05, see Figure 4.



*τ*1=-2.95

*Figure 4: Pictorial representation of a thresholds () that divides an underlying continuous latent response variable into two categories with proportions 0.05 and 0.95 respectively.*

In the **MODEL:** command of Mplus, variables within brackets [ ] refer to the *variable means (if interval variables*) or to *variable thresholds in the case of categorical variables*.

The different thresholds of a categorical variable are labelled using the Dollar sign **$** followed by the threshold number. Since variable **ranti86** has 3 response categories, it will have 2 thresholds indicated by:

[ranti86 $1];

[ranti86 $2];

The star sign **\*** is used to free a parameter: however, if **\*** is followed by a number, the estimation of that parameter will have a start value equal to the number specified after **\***. For example, to provide starting value of -1 for first threshold of **ranti86**, I would write:

[ranti86 $1\*-1];

The @ symbol fixes a parameter at a user-specified value. For example, to fix the second threshold of ranti86 to 2, I would write:

[ranti86 $2@2];

Parentheses are used to name or to constrain a parameter. Names are provided when letters are within parentheses. E.g. to name the threshold #1 of variable **ranti86** as “p1” and that of variable ranx86 as “p2”:

[ranti86 $1] (p1);

[ranx86 $1] (p2);

These parameters can then be constrained using the **MODEL CONSTRAINT:** option in the **MODEL:** command. For example, to impose equality constraints on these two thresholds for variable **ranti86** and ranx86:

**MODEL CONSTRAINT: p1 = p2**;

The same equality constraint can be imposed using numbers between parentheses: if thresholds of variables *a* and *b* are constrained equal, write:

[ranti86 $1] (1);

[ranx86$1] (1);

Therefore, to ensure that we estimate the associations of latent variable **x** with indicators collected in 1986 and that of latent variable **y** with indicators collected in 1988, we will have to specify this:

VARIABLES:

NAMES = […] ;

MISSING= all (-999);

USEVAR=ranti86 ranx86 rhypr86 rdep86 rpeer86

ranti88 ranx88 rhypr88 rdep88 rpeer88;

CATEGORICAL=ranti86 ranx86 rhypr86 rdep86 rpeer86

ranti88 ranx88 rhypr88 rdep88 rpeer88;

CLASSES= x(2) y(3);

ANALYSIS:

TYPE=mixture;

**MODEL:**

%OVERALL%

y#1 ON x#1;

y#2 ON x#1;

**MODEL x:**

**%x#1%**

[ ranti86$1\*-1 .5];

[ ranti86$2\*-1 ];

[ ranx86$1\*-1.5 ];

[ ranx86$2\*-1 ];

[ rhypr86$1\*-1. ];

[ rhypr86$2\*-1 ];

[ rdep86$1\*-1 .5];

[ rdep86$2\*-1 ];

[ rpeer86$1\*-1.5 ];

[ rpeer86$2\*-1 ];

**%x#2%**

[ ranti86$1\*1 ];

[ ranti86$2\*1 .5];

[ ranx86$1\*1 ];

[ ranx86$2\*1.5 ];

[ rhypr86$1\*1 ];

[ rhypr86$2\*1.5 ];

[ rdep86$1\*1 ];

[ rdep86$2\*1.5 ];

[ rpeer86$1\*1 ];

[ rpeer86$2\*1.5 ];

**MODEL y:**

**%y#1%**

[ ranti88$1\*-1.5 ];

[ ranti88$2\*-1];

[ ranx88$1\*-1.5 ];

[ ranx88$2\*-1 ];

[ rhypr88$1\*-1.5 ];

[ rhypr88$2\*-1 ];

[ rdep88$1\*-1.5];

[ rdep88$2\*-1 ];

[ rpeer88$1\*-1.5];

[ rpeer88$2\*-1 ];

**%y#2%**

[ ranti88$1\*0 ];

[ ranti88$2\*0.5];

[ ranx88$1\*0 ];

[ ranx88$2\*0.5];

[ rhypr88$1\*0 ];

[ rhypr88$2\*0.5 ];

[ rdep88$1\*0];

[ rdep88$2\*0.5 ];

[ rpeer88$1\*0];

[ rpeer88$2\*0.5 ];

**%y#3%**

[ ranti88$1\*1 ];

[ ranti88$2\*1.5];

[ ranx88$1\*1 ];

[ ranx88$2\*1.5];

[ rhypr88$1\*1 ];

[ rhypr88$2\*1.5 ];

[ rdep88$1\*1];

[ rdep88$2\*1.5 ];

[ rpeer88$1\*1];

[ rpeer88$2\*1.5 ];

The section of the model:

**MODEL x:**

**%x#1%**

[ ranti86$1\*-1 .5];

[ ranti86$2\*-1 ];

[ ranx86$1\*-1.5 ];

[ ranx86$2\*-1 ];

…

is instructing Mplus to estimate the thresholds of indicator **ranti86** in latent class 1 of **x**, the thresholds of indicator **ranx86** in latent class 1 of **x**, and so on. Note that since these indicators have three categories, there are two thresholds. Furthermore, note that the starting values at which the thresholds are located are increasing from threshold 1 to threshold 2 (e.g.:

[ ranti86$1\*-1 .5];

[ ranti86$2\*-1 ];

The reason why thresholds should be located at increasing values is evident when inspecting Figure 2 above: Thresholds are located at increasing values of the underlying continuum that “cuts off” the underlying variable.

By attributing different starting values to the indicator thresholds in latent class 1 and 2 of **x**, we are instructing Mplus to start by identifying a latent class with higher probability of high scores in the indicators and another one with lower probability of high scores in the indicators.

Note that all the measurement parameters of latent classes of latent variable **x** involve indicators collected in 1986. Conversely, all the measurement parameters of latent variables **y** involve indicators collected in 1988. In this way, we are instructing Mplus to consider latent variable **x** as the explanatory variable for indicators in 1986, whereas latent variable **y** is going to be considered as the explanatory variable of indicators collected in 1988. In this way, we are specifying the model depicted in Figure 1.

## **Constrain Measurement Parameters**

We can use the syntax displayed so far to constrain measurement parameters across latent classes.

Firstly, when running LTA, we may test models with *full measurement invariance,* provided the latent classes we have identified at each time point are similar.

Let’s assume we estimated two latent classes based on the indicators collected by NLS in 1986 and in 1988. If I wanted to constrain the measurement parameters of these latent variables to be the same across time, I will write the model in a similar manner:

VARIABLES:

NAMES = […] ;

MISSING= all (-999);

USEVAR=ranti86 ranx86 rhypr86 rdep86 rpeer86

ranti88 ranx88 rhypr88 rdep88 rpeer88;

CATEGORICAL=ranti86 ranx86 rhypr86 rdep86 rpeer86

ranti88 ranx88 rhypr88 rdep88 rpeer88;

**CLASSES= x(2) y(2);**

ANALYSIS:

TYPE=mixture;

**MODEL:**

%OVERALL%

y#1 ON x#1;

y#2 ON x#1;

**MODEL x:**

**%x#1%**

[ ranti86$1\*] (x1a1);

[ ranti86$2\*] (x1a2) ;

[ ranx86$1\*] (x1b1);

[ ranx86$2\*] (x1b2);

[ rhypr86$1\*] (x1c1);

[ rhypr86$2\*] (x1c2);

[ rdep86$1\*] (x1d1);

[ rdep86$2\*] (x1d2);

[ rpeer86$1\*] (x1e1);

[ rpeer86$2\*] (x1e2);

**%x#2%**

[ ranti86$1\*] (x2a1);

[ ranti86$2\*] (x2a2);

[ ranx86$1\*] (x2b1);

[ ranx86$2\*] (x2b2);

[ rhypr86$1\*] (x2c1);

[ rhypr86$2\*] (x2c2);

[ rdep86$1\*] (x2d1);

[ rdep86$2\*] (x2d2);

[ rpeer86$1\*] (x2e1);

[ rpeer86$2\*] (x2e2);

**MODEL y:**

**%y#1%**

[ ranti88$1\* ] (x1a1);

[ ranti88$2\*] (x1a2) ;

[ ranx88$1\* ] (x1b1);

[ ranx88$2\* ] (x1b2);

[ rhypr88$1\*] (x1c1);

[ rhypr88$2\*] (x1c2);

[ rdep88$1\*] (x1d1);

[ rdep88$2\*] (x1d2);

[ rpeer88$1\*] (x1e1);

[ rpeer88$2\* ] (x1e2);

**%y#2%**

[ ranti88$1\* ] (x2a1);

[ ranti88$2\*] (x2a2);

[ ranx88$1\* ] (x2b1);

[ ranx88$2\* ] (x2b2);

[ rhypr88$1\*] (x2c1);

[ rhypr88$2\*] (x2c2);

[ rdep88$1\*] (x2d1);

[ rdep88$2\*] (x2d2);

[ rpeer88$1\*] (x2e1);

[ rpeer88$2\*] (x2e2);

|  |  |
| --- | --- |
|  |  |

The strings and numbers between parentheses are providing names to identify the thresholds in each line. Note that thresholds in latent class 1 of **x** and in latent class of **y** are given the same names (e.g. **x1a1**):

**%x#1%**

[ ranti86$1\*] (x1a1);

[ ranti86$2\*] (x1a2) ;

[ ranx86$1\*] (x1b1);

[ ranx86$2\*] (x1b2);

[ rhypr86$1\*] (x1c1);

[ rhypr86$2\*] (x1c2);

[ rdep86$1\*] (x1d1);

[ rdep86$2\*] (x1d2);

[ rpeer86$1\*] (x1e1);

[ rpeer86$2\*] (x1e2);

…

**%y#1%**

[ ranti88$1\* ] (x1a1);

[ ranti88$2\*] (x1a2) ;

[ ranx88$1\* ] (x1b1);

[ ranx88$2\* ] (x1b2);

[ rhypr88$1\*] (x1c1);

[ rhypr88$2\*] (x1c2);

[ rdep88$1\*] (x1d1);

[ rdep88$2\*] (x1d2);

[ rpeer88$1\*] (x1e1);

[ rpeer88$2\* ] (x1e2);

In this way, we are ensuring that the first threshold of **ranti86** in latent class 1 of **x**, will have the same value as the first threshold of **ranti88** in latent class 1 of **y**, that is, the two variables will have the same measurement parameters (i.e. the same response probabilities conditional on latent class affiliation). In other words, we are imposing equality constraints that ensure the associations between latent classes and indicators are constant across time points.

Other possible constraints on response items can be imposed considering that the conditional probability of an item response (e.g. **ranti86** = 0) given a latent class value (e.g. **x** = 1) is given by the inverse-logit function:

**ranti86** **=0 | x = 1) =**

where represents the first threshold of variable **ranti86** given affiliation to class 1 of the latent variable **x.**

Therefore, if we wanted to constrain the probability of reporting the lowest score in variable **ranti86** for those in latent class 1 of **x** to be virtually 0, we would assign a very low value to the first threshold of this variable in latent class 1, e.g. -15:

MODEL:

…

MODEL x:

%x#1%

**[ ranti86** **$1@-15 ];**

[ ranti86 $2 \*-1 ];

**…**

The wildcard symbol @ fixes the threshold to the logit value = -15, which corresponds to a very low, virtually 0, probability of reporting the lowest score in the variable for those in latent class 1 of latent variable **x.**

## **Specify Measurement Parameters for Continuous Indicators**

Latent Class and Latent Transition models are often applied to categorial indicators. However, they can be easily applied to any type of indicators, including continuous ones.

Assume the indicators we are working with were four memory tests **memo1**, **memo2**, **memo3**, and **memo4** with standardised scores ranging from -3 to +3. If we wanted to specify that latent classes of “memo\_cl” should capture individuals with scores below 0 and scores above 0 respectively, I could have written the model like this:

VARIABLES:

NAMES = […] ;

MISSING= all (-999);

USEVAR=memo1 memo2 memo3 memo4;

CLASSES= memo\_per(2);

ANALYSIS:

TYPE=mixture;

**MODEL:**

%OVERALL%

**%memo\_per#1%**

[memo1\*-1.5];

[memo2\*-1.5];

[memo3\*-1.5];

[memo4\*-1.5];

**%memo\_per#2%**

[memo1\*1.5];

[memo2\*1.5];

[memo3\*1.5];

[memo4\*1.5];

The **[memo1\*-1.5]**, in latent class 1 of latent variable “**memo\_per**” **%memo\_per#1%**, specifies that the starting value of the average performance in **memo1**, **memo2**, etc. for those in latent class 1 will be a negative value equal to -1.5, whereas the average performance in the memory tests for those in latent class 2 will be positive.

If I wanted to specify that individuals in latent class 1 of **memo\_per** should have the same average performance across the tests, I would write a similar command:

**MODEL:**

%OVERALL%

**%memo\_per#1%**

[memo1\*-1.5] (1);

[memo2\*-1.5] (1);

[memo3\*-1.5] (1);

[memo4\*-1.5] (1);

**%memo\_per#2%**

[memo1\*1.5];

[memo2\*1.5];

[memo3\*1.5];

[memo4\*1.5];

The **(1)** is instructing Mplus to fix the average of **memo1**, **memo2**, etc. to be the same for individuals in latent class 1 of latent class variable **memo\_per**.

Using the same principles, we could specify a model where we constrain the variances of all the indicators to be the same across latent classes:

**MODEL:**

%OVERALL%

**%memo\_per#1%**

[memo1\*-1.5] ;

**memo1 (v1);**

[memo2\*-1.5] ;

**memo2 (v2);**

[memo3\*-1.5] ;

**memo3 (v3) ;**

[memo4\*-1.5] ;

**memo4 (v4);**

**%memo\_per#2%**

[memo1\*1.5];

**memo1 (v1);**

[memo2\*1.5];

**memo2 (v2);**

[memo3\*1.5];

**memo3 (v3);**

[memo4\*1.5];

**memo4 (v4);**

The **memo1**, **memo2**, etc. is referring to the variance of the indicators. By following these variables with **(v1)**, etc., we are giving these parameters a name. By giving the same name **(v1)** to the variance of **memo1** in latent class 1 and in latent class 2, we are instructing Mplus to consider these parameters to be the same across latent classes.