

Introduction to Latent Class 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:

- `ess_ex1.dat`

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 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.

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 which 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 (text is free).

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\data1.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\data1.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 = essround idno polintr rpsppsgv ractrolg rpsppi1 rcptppol rptcpplt retapapl;

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' views, write:

```
USEVARIABLES = rpsppsgv ractrolg rpsppi1 rcpttpol rptcpplt retapapl;
```

This will ensure other variables, e.g. IDs, will not be 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. For example, variable "cou" (The variable indicating respondents' country) should not be listed under **CATEGORICAL** since it is not going to be used as a dependent variable in the model.

e.g.:

```
CATEGORICAL = rpsppsgv ractrolg rpsppi1 rcpttpol rptcpplt retapapl;
```

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 = ess7id;
```

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 country level:

```
CLUSTER=cou;
```

*!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);
```

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 constraints 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 0 of the x-axis, the second appearing at point 1 of the x-axis, etc.:

```
PLOT: Type = PLOT3;
SERIES= rpsppsgv(0) ractrolg(1) rpsppi(2) rcptppol(3) rptcpplt(4) retapapl(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= rpsppsgv(1) ractrolg(1.5) rpsppi(2) rcptppol(2.5) rptcpplt(3) retapapl(3.5);
```

SAVEDATA:

Instructs the programme to create a file with parameters estimated.

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

```
FILE IS analysis1.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 variable and the number of classes estimate.

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

VARIABLES:

```
NAMES = essround idno polintr rpsppsgv ractrolg rpsppipl rcpttpol rptcpplt retapapl;
USEVAR = rpsppsgv ractrolg rpsppipl rcpttpol rptcpplt retapapl;
CATEGORICAL = rpsppsgv ractrolg rpsppipl rcpttpol rptcpplt retapapl;
MISSING = all (-999);
IDVARIABLE = ess7id;
```

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

VARIABLES:

```
NAMES = essround idno polintr rpsppsgv ractrolg rpsppipl rcpttpol rptcpplt retapapl;
USEVAR = rpsppsgv ractrolg rpsppipl rcpttpol rptcpplt retapapl;
CATEGORICAL = rpsppsgv ractrolg rpsppipl rcpttpol rptcpplt retapapl;
MISSING = all (-999);
IDVARIABLE = ess7id;
CLASSES=c(2);
```

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

VARIABLES:

```
NAMES = essround idno polintr rpsppsgv ractrolg rpsppipl rcpttpol rptcpplt retapapl;
USEVAR = rpsppsgv ractrolg rpsppipl rcpttpol rptcpplt retapapl;
CATEGORICAL = rpsppsgv ractrolg rpsppipl rcpttpol rptcpplt retapapl;
MISSING = all (-999);
IDVARIABLE = ess7id;
```



```
CLASSES=activity(6);
```

NOTE: If you want to estimate models with different number of latent classes, say 2 to 8, you will have to write separate INPUT files for each model with different classes. This operation can be tedious, as is the operation of extracting the key statistics from the large Mplus OUTPUT files. If you use R, the suit MplusAutomation (Hallquist, M. N. & Wiley, J. F. (2018). MplusAutomation: An R Package for Facilitating Large-Scale Latent Variable Analyses in Mplus. Structural Equation Modeling, 25, 621-638. doi: 10.1080/10705511.2017.1402334) can facilitate these tasks.

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 = essround idno polintr rpsppsgv ractrolg rpsppipl rcptppol rptcpplt retapapl;
```

```
USEVAR = rpsppsgv ractrolg rpsppipl rcptppol rptcpplt retapapl;
```

```
CATEGORICAL = rpsppsgv ractrolg rpsppipl rcptppol rptcpplt retapapl;
```

```
MISSING = all (-999);
```

```
IDVARIABLE = ess7id;
```

```
CLASSES=activity(6);
```

```
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.

For example, adding these lines:

```
VARIABLES:
```

```
NAMES = essround idno polintr rpsppsgv ractrolg rpsppipl rcptppol rptcpplt retapapl;
```

```
USEVAR = rpsppsgv ractrolg rpsppipl rcptppol rptcpplt retapapl;
```

```
CATEGORICAL = rpsppsgv ractrolg rpsppipl rcptppol rptcpplt retapapl;
```

```
MISSING = all (-999);
```

```
IDVARIABLE = ess7id;
```

```
CLASSES=activity(6);
```

```
CLUSTER=cou;
```

ANALYSIS:

```
TYPE=mixture complex;
```

ensures that estimation of the models controls for the clustering at the variable “cou” (Country) level.

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 Latent Class Model Parameters

If you are conducting exploratory latent class analysis (i.e. you want to test models with different number of classes) there is no need to define latent class parameters in the **MODEL:** command.

The **MODEL:** command however gives you the possibility of imposing constraints on the model you are estimating, with the added possibility of performing formal tests (e.g. likelihood ratio tests) to check differences between the model with the added constraint and the nested model that includes the same parameters, save for the added ones.

To understand how the model parameters work, we need to consider 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 expressed as logits (log-odds)

To clarify, consider Figure 1:

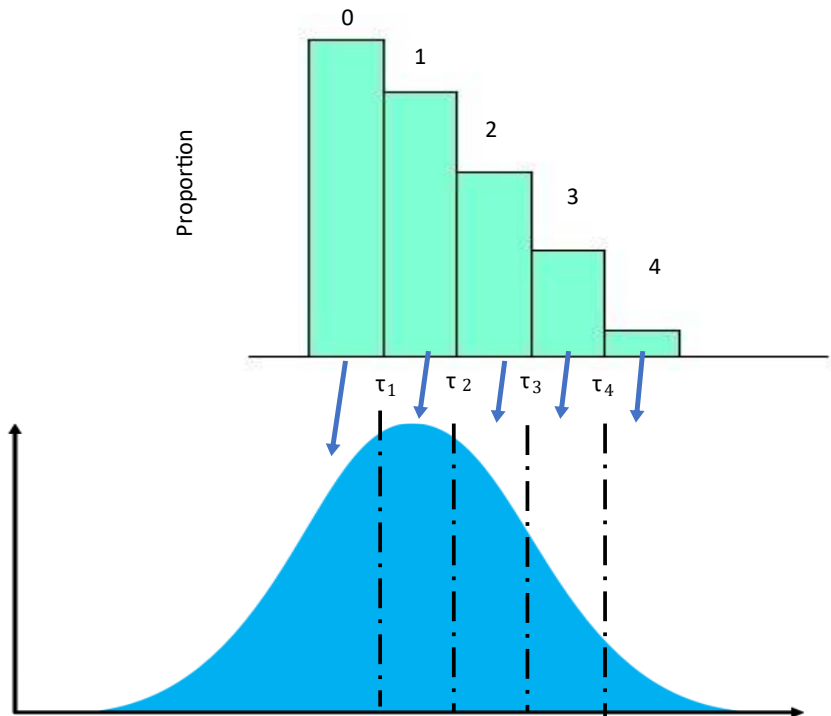


Figure 1: Pictorial representation of how thresholds (τ_s) in an underlying continuous latent response variable correspond to the categories in an ordered categorical observed variable.

Figure 1 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 (τ_s). 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 (τ_s). For example, had we placed the 4th threshold (τ_4) at a higher level on the x-axis, we would have fewer or no cases into category 4 of the observed categorical variable.

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 \$ followed by the threshold number. If variable **rspsgv** has 3 response categories, it will have 2 thresholds indicated by:
[rspsgv \$1];
[rspsgv \$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 **rspsgv**, you would write:

[rspsgv \$1*-1];

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

```
[rspppsgv $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 thresholds #1 of variable **rspppsgv** as “p1” and that of variable **ractrolg** as “p2”:

```
[rspppsgv$1] (p1);  
[ractrolg$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 **rspppsgv** and **ractrolg**:

```
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:

```
[rspppsgv$1] (1);  
[ractrolg$1] (1);
```

Therefore, when we estimate a model with two latent classes like this:

```
VARIABLES:  
NAMES = essround idno polintr rspppsgv ractrolg rspppi1 rcpttpol rptcpplt retapap1;  
USEVAR = rspppsgv ractrolg rspppi1 rcpttpol rptcpplt retapap1;  
CATEGORICAL = rspppsgv ractrolg rspppi1 rcpttpol rptcpplt retapap1;  
MISSING = all (-999);  
IDVARIABLE = ess7id;  
CLASSES=c(2);
```

What MPlus is doing is estimating the thresholds of each indicator within each latent class, as well as the “mean” that defines the division between the two classes estimated.

We can see these parameters if we invoke option **SVALUES** (i.e. Starting values) in command **OUTPUT:** and check the OUTPUT file to look for information on “Model Command With Final Estimates Used as Starting Values”:

```
%OVERALL%  
[ class#1*0.43670 ];  
%CLASS#1%  
[ rspppsgv$1*1.16027 ];  
[ rspppsgv$2*3.18564 ];  
[ ractrolg$1*0.92750 ];  
[ ractrolg$2*2.27577 ];  
[ rspppi1$1*1.65511 ];
```

```
[ rpsppi1$2*4.27391 ];
[ rcptppol$1*0.46739 ];
[ rcptppol$2*1.87147 ];
[ rptcpplt$1*1.36123 ];
[ rptcpplt$2*3.83421 ];
[ retapapl$1*1.09907 ];
[ retapapl$2*3.08364 ];
%CLASS#2%
[ rpsppsgv$1*-1.63560 ];
[ rpsppsgv$2*0.63602 ];
[ ractrolg$1*-1.30507 ];
[ ractrolg$2*0.47925 ];
[ rpsppi1$1*-2.05611 ];
[ rpsppi1$2*0.77792 ];
[ rcptppol$1*-2.07007 ];
[ rcptppol$2*0.29277 ];
[ rptcpplt$1*-1.22186 ];
[ rptcpplt$2*1.33150 ];
[ retapapl$1*-1.53791 ];
[ retapapl$2*0.76390 ];
```

The example above also shows two other characteristics of the **MODEL:** syntax for latent classes.

How to Specify Latent Class Parameters in LCA

When we are writing the **MODEL:** syntax, we have start by specifying parameters that are valid for all classes, which are listed after the **%OVERALL%** line.

For example, if we wanted to estimate the regression of the latent classes estimated on a covariate “male”, we would write this regression after the **%OVERALL%** line like this:

```
VARIABLES:
NAMES = essround idno polintr rpsppsgv ractrolg rpsppi1 rcptppol rptcpplt retapapl;
USEVAR = rpsppsgv ractrolg rpsppi1 rcptppol rptcpplt retapapl male;
CATEGORICAL = rpsppsgv ractrolg rpsppi1 rcptppol rptcpplt retapapl;
MISSING = all (-999);
IDVARIABLE = ess7id;
CLASSES=c(2);

ANALYSIS:
TYPE=mixture;

MODEL:
%OVERALL%
[class#1*0.43670];
class ON male;
```

Note that the covariate “male” has been added to the list of variables being used in analyses after **USEVAR=** in option **VARIABLES:** command, as this is necessary for Mplus to include this as a covariate. (Note also that independent variables are not defined as **CATEGORICAL=**, since this information is only necessary for dependent variables).

After the **%OVERALL%** option, it is possible to *specify specific parameters for the different classes* of the latent categorical variable. In the example above, we specified two latent classes. The class-specific parameters are specified after lines:

```
%CLASS#1%  
%CLASS#2%
```

What follows after **%CLASS#1%** will be parameters that apply only to category 1 of latent class “Class”, and what follows after **%CLASS#2%** are parameters that pertain only to category 2 of latent class “Class”.

To be clear, if our latent class was called “activity” rather than “class” and had 2 classes, the statement in **MODEL:** will be written as:

```
MODEL:  
%OVERALL%  
%ACTIVITY#1%  
%ACTIVITY#2%
```

Note also that **%ACTIVITY#1%** and **%ACTIVITY#2%** are not followed by semicolon.

If we call our latent variable “activity” and estimate it has two latent classes, the line **%ACTIVITY#1%** (indicating latent classes 1 of the latent variable) are class-specific starting values of the indicators thresholds. These starting value differ for latent class 2, as you can see after the line **%ACTIVITY#2%:**

```
MODEL:  
%OVERALL%  
[ class#1*0.43670 ];  
%ACTIVITY#1%  
[ rpsppsgv$1 *1.16027 ];  
[ rpsppsgv$2 *3.18564 ];  
...  
%ACTIVITY#2%  
[ rpsppsgv$1 *-1.63560 ];  
[ rpsppsgv$2 *0.63602 ];
```

In order to understand how the thresholds work, we need to consider they are expressed using the logit of a probability, whereby:

$$\text{logit}(p) = -\log\left(\frac{1}{p} - 1\right);$$

Logits range from $-\infty$ to ∞ . Using the formula above, a probability = .50 corresponds to $\text{logit}=0$. 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 2.

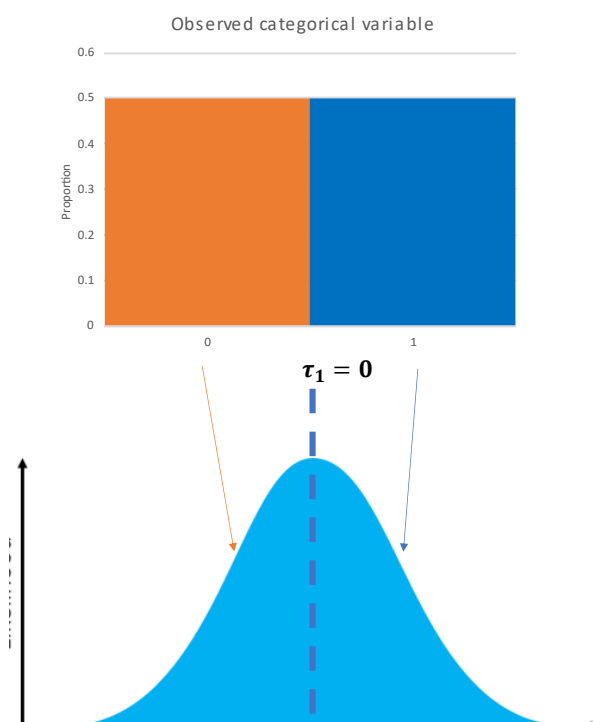


Figure 2: Pictorial representation of a thresholds (τ_1) that divides an underlying continuous latent response variable into two categories with the same proportion.

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 3.

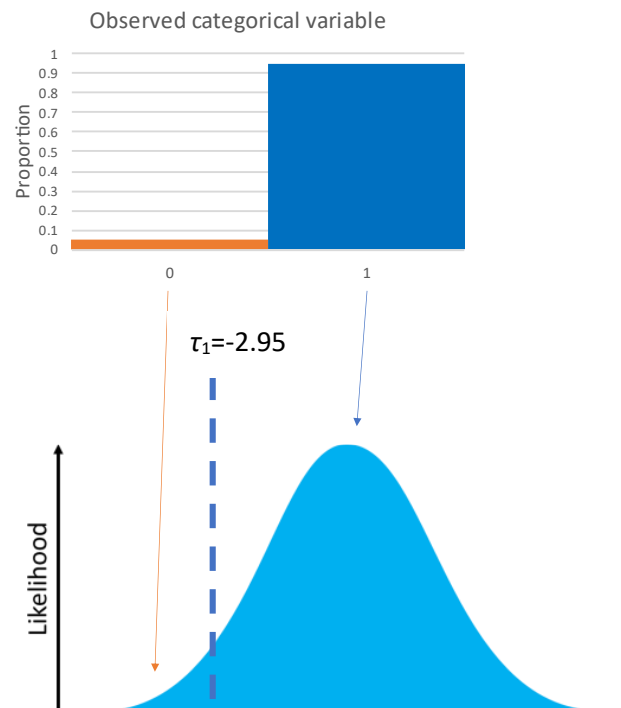


Figure 3: Pictorial representation of a thresholds (τ_1) that divides an underlying continuous latent response variable into two categories with proportions 0.05 and 0.95 respectively.

The conditional probability of an item response (e.g. **rpsppsgv** = 1) given a latent class value (e.g. **activity** = 1) is given by the inverse-logit function:

$$p(\mathbf{rpsppsgv} = 1 \mid \mathbf{activity} = 1) = \frac{1}{1 + \exp(-\tau_{\mathbf{rpsppsgv}=1 \mid \mathbf{activity}=1})}$$

where $\tau_{\mathbf{rpsppsgv}=1 \mid \mathbf{activity}=1}$ represents the first threshold of variable **rpsppsgv** given affiliation to class 1 of the latent variable **activity**.

Therefore, if we, for example, wanted to constrain the probability of reporting the lowest score in variable **rpsppsgv** for those in latent class 1 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:
%OVERALL%
[ class#1*0.43670 ];
%ACTIVITY#1%
[ rpsppsgv$1@-15 ];
[ rpsppsgv$2 *3.18564 ];
...
```

Following this command, the output will show that the probability of reporting the lowest score in the variable is estimated being equal to 0:

RESULTS IN PROBABILITY SCALE

Latent Class 1

RPSPPSGV



Category 1	0.000	0.000	0.000	1.000
Category 2	0.563	0.110	5.093	0.000
Category 3	0.437	0.110	3.958	0.000

RACTROI/G