Models with Discrete Latent Variables for Analysis of Categorical Data: A MATLAB MDLV toolbox

SUPPLEMENTAL MATERIALS

Appendix A

EVS Data Set

Data sampled from the European Value Study (EVS) are used to illustrate the usage of the MDLV toolbox. Ten items (cf. Table 1) from the section of studying the attitude of the important criteria for a successful marriage are analyzed. We sampled 50 participants from each of the 31 countries in Wave 3 (1999) and Wave 4 (2008). Responses to the 10 items of the sampled participants are analyzed separately for Wave 3 and Wave 4 using a LCM and a MLCM.

A synthetic longitudinal data set was constructed for fitting LMM and MLMM with the MDLV toolbox. The background information available in the EVS database was used to link each respondent in Wave 3 to a demographically similar one in Wave 4 to yield a matched pair. Responses to the items of each pair were then treated as repeated responses over the two waves by a same individual. Country and household income (variable x047) were used as the matching criteria. Operationally, each participant within a country was ranked by household income for each wave. Then within each country, individuals of the same rank in the two waves were paired. The resulting data set is referred to as 'synthetic longitudinal data' hereafter.

The EVS data were available in the SPSS format. For the purpose of sampling data to be used in the present analysis, the SPSS data were saved in SAS format for later processing. A 3-point rating scale was used for each item in the survey: very important, rather important, and not very important. As the rating distributions for the items were skewed with most responses indicating 'very important', the other two ratings were combined into one for subsequent analysis. The 'very important' rating was recoded to '1' while 'rather important' and 'not very important' ratings were recoded to '0'. For further preparation of data for use in the MDLV toolbox, the SAS program was employed to sample participants, link data between waves, and recode item responses. Appendix A1 presents the SAS codes for creating the base data sets. Appendix A2 provides the details of the MATLAB syntax for preparing the data sets to fit LCM, LMM, MLCM, and MLMM using the MDLV toolbox.

Appendix A1: SAS Codes for data sampling, linking, and recoding

```
options pagesize=10000 linesize=256 nodate nocenter nonumber;
libname WVS 'C:\Documents and Settings\Administrator\Desktop\WVS2005';

Data Job1; set WVS.WVS_include3;RUN;
PROC CONTENTS; RUN;
/*
d027     important in marriage: faithfulness
d028     important in marriage: adequate income
d029     important in marriage: same social background
d031     important in marriage: shared religious beliefs
d032     important in marriage: good housing
```

```
d033     important in marriage: agreement on politics
d035     important in marriage: live apart from in-laws
d036     important in marriage: happy sexual relationship
d037     important in marriage: share household chores
d038     important in marriage: children
 2    s003        Num       8    country code
 3    s006        Num       8    original respondent number
 4    s007        Num       8    unified respondent number
 5    s009        Char      9    country abbreviation
 6    s020        Num       8    survey year
 1    s002evs     Num       8    EVS-wave
44    x047        Num       8    income household respondent
*/

/*  Sampling */
Data KeepWave34; SET Job1; *Delete country without complete data for
both Wave 3 and Wave 4;
IF s003 in(8,31,51,70,196,197,268,498,499,688,756,807,915,124,578,840)
then delete;
IF s002evs in(1,2) then delete; *Delete wave 1 and wave 2;
array Marriage[10] d027 d028 d029 d031 d032 d033 d035 d036 d037 d038;
Do i= 1 to 10; if Marriage(i)<0 then delete;END;
RUN;

PROC SORT data=KeepWave34;
    BY s003 s002evs;
RUN;

* sample 50 subjects for each country and each wave.;
PROC SURVEYSELECT data=KeepWave34
        method=srs n=50
        seed=123 out=SampleN50;
          strata s003 s002evs;
RUN;

Data SampleN50_Marriage(keep=s003 s002evs x047 d027 d028 d029 d031 d032
d033 d035 d036 d037 d038);
set SampleN50;
* recode the data;
Data SampleN50_MarriageV2; set SampleN50_Marriage;
array Marriage[10] d027 d028 d029 d031 d032 d033 d035 d036 d037 d038;
array M[10] M1-M10;
Do i= 1 to 10; if Marriage(i) in(2,3) then M(i)=1; else M(i)=0;END;
RUN;

Data SampleN50_MarriageV3; set SampleN50_MarriageV2;
IF s003 in(191,792) then delete; *Delete wave 1 and wave 2;
RUN;

* Link both waves through x047: household income;
PROC SORT data=SampleN50_MarriageV3;
    BY s003 s002evs x047;
```

```
RUN;
/*PROC PRINT data=SampleN50_MarriageV3; RUN;*/

Data IDindex;
     Do Country=1 to 31;
          Do Wave=1 to 2;
               Do Subject=1 to 50;
                    output;
               End;
          End;
     End;
RUN;
PROC PRINT data=IDindex; RUN;

Data SampleN50_MarriageV4; merge IDindex SampleN50_MarriageV3;RUN;

PROC PRINT data= SampleN50_MarriageV4;
var Country Subject Wave M1-M10;
RUN;
```

Appendix A2: Data Preparation MATLAB Codes

There are 13 variables in the data set (Data), the names and order the variables are: "Country" "Subject" "Wave" "M1" "M2" "M3" "M4" "M5" "M6" "M7" "M8" "M9" "M10." The first three variables indicate the group ID, participant ID, and indication of waves for the responses of the 10 items in column 4 to 13. This information was used to reshape the responses of the 10 items into appropriate dimensions for specific models. The matrix (`Data2D`) includes all responses of the J (=10) items as columns, and rows are combinations of group ID, participant ID, and indication of waves. This matrix has the dimension of ($G \times Ng \times T$) by J. In this particular example, there are 3100 ($31 \times 50 \times 2$) rows and 10 columns. Note that G (=31) is the number of countries, N_g (=50) is the number of subjects, and T (=2) is the number of time points (survey waves).

This 2D data (`Data2D`) can be restructured into the required 4D data using the MATLAB function of *reshape* and *permute* with the dimension $G \times Ng \times J \times T$ for fitting MLMM (`Data4D`). The data used for fitting MLCM were obtained by separating the synthetic longitudinal data into two time waves (`ItemsT1_3D` and `ItemsT2_3D`), each is a $G \times Ng \times J$ matrix. The LCM and LMM can be applied when the nested data structure is ignored in the analyses. Thus a $N \times J \times T$ matrix (Data3D) is required for fitting LMM, where N ($= G \times Ng$) is the total sample size (=1550 in this example). For each time point, a $N \times J$ data matrix is required to fit with LCM (`ItemsT1_2D` and `ItemsT2_2D`). The EVSdata.m script contains the syntax to create data sets used in the analyses with these models. Executing this file will produce all the required data sets for the illustrative examples in this paper.

EVSdata.m MATLAB codes:

```
clc
```

```
clear

Data=[
    1          1       1      1     1     0     0     1     0     0     1     0     1 ;
    1          1       2      1     1     0     0     1     0     1     0     0     0 ;
    1          2       1      0     0     0     1     0     0     1     1     0     0 ;

...

   31         49       2      1     1     0     1     0     1     0     1     0     0 ;
   31         50       1      1     0     0     1     1     0     0     1     0     0 ;
   31         50       2      1     1     0     0     0     0     1     1     1     0 ];

%% Data matrix (4D) for MLMM (G X Ng X J X T)
Data2D=Data(:,4:13); %G*ng*t J keep only 10 items
A1=reshape(Data2D',[10,2,50,31]); % transform into 4D data matrix.
Data4D=permute(A1,[4,3,1,2]);  %G,Ng,J,T
%% reshape data matrix into 3D for LMM (N X J X T);
%Data4D : G,Ng,J,T
Data3D=reshape(Data4D,[50*31,10,2]); %(N X J X T)

%% Data matrix (2D) for LCM (G X Ng , J)
% data for time 1
idxT1 = ( Data(:,3)==1 );
ItemsT1_2D = Data(idxT1,:); %(G X Ng , J);
ItemsT1_2D = ItemsT1_2D(:,4:13);
% data for time 2
idxT2 = ( Data(:,3)==2 );
ItemsT2_2D = Data(idxT2,:); %(G X Ng , J);
ItemsT2_2D = ItemsT2_2D(:,4:13);

%% reshape data matrix into 3D for MLCM (G X Ng X J)
% wave=3;
ItemsT1=ItemsT1_2D; % Keep items only;
ItemsT13D=reshape(ItemsT1',[10,50,31]); %J X Ng X G;
ItemsT1_3D=permute(ItemsT13D,[3,2,1]); %G X Ng X J;
% wave=4;
ItemsT2=ItemsT2_2D; % Keep items only;
ItemsT23D=reshape(ItemsT2',[10,50,31]); %J X Ng X G;
ItemsT2_3D=permute(ItemsT23D,[3,2,1]); %G X Ng X J;

%% Summary: Data for
%LCM:  Wave3:ItemsT1_2D
%      Wave4:ItemsT2_2D
%LMM:  Data3D
%MLCM: Wave3:ItemsT1_3D
%      Wave4:ItemsT2_3D
%MLMM: Data4D
```

Appendix B

Appendix B1

Fitting MLMM using the MDLV MATLAB toolbox (cf. Table 2, Table 3, and Table 4)

The script for fitting MLMM is included in 'EVSdata_MLMM.m' as listed below. The required 4D data (Data4D) is created earlier (see Appendix A2). Four models are fitted to the data by specifying the number of clusters to be 2 or 3 and the number of classes to be 2 or 3. The required starting values are randomly generated using the function 'StartingValue_RandomV1'. The main function to estimate the parameters of the MLMM is 'MLMM.m'. The subfunction 'Loglikelihood_MLMM.m' is required to compute the likelihood value at each iteration.

EVSdata_MLMM.m

```
%% ======   Fit MLMM   ======
%% === 2-clutsers 2-Classes MLMM ===
% Specify the parameters:
G = 31; % Number of groups
Ng= 50; % Number of subjects per group
J=10;   % Number of items
T=2;    % Number of time points
L = 2;  % Number of latent clusters
M = 2;  % Number of latent classes

% generate the starting values
[PH_g,PXgivenH,TauGivenH,TauGivenHT,CondResProd]=StartingValue_RandomV1
(1,M,L,T,J);

%  Fit the MLMM
[lk,iteration,G,Ng,J,T,L,M,PH_g,PXgivenH,TauGivenH,CondResProd,Est_h,np
,AIC,BIC]=...
    MLMM(Data4D,Data2D,G,Ng,L,M,J,T,PH_g,PXgivenH,TauGivenH,CondResProd)
%% === 2-clutsers 3-Classes MLMM ===
% Specify the parameters:
G = 31; Ng= 50; J=10;  T=2;
L = 2;  % Number of latent clusters
M = 3;  % Number of latent classes

% generate the starting values
[PH_g,PXgivenH,TauGivenH,TauGivenHT,CondResProd]=StartingValue_RandomV1
(1,M,L,T,J);

%  Fit the MLMM
[lk,iteration,G,Ng,J,T,L,M,PH_g,PXgivenH,TauGivenH,CondResProd,Est_h,np
,AIC,BIC]=...
    MLMM(Data4D,Data2D,G,Ng,L,M,J,T,PH_g,PXgivenH,TauGivenH,CondResProd)

%% === 3-clutsers 2-Classes MLMM ===
```

```
% Specify the parameters:
G = 31; Ng= 50; J=10;  T=2;
L = 3;  % Number of latent clusters
M = 2;  % Number of latent classes

% generate the starting values
[PH_g,PXgivenH,TauGivenH,TauGivenHT,CondResProd]=StartingValue_RandomV1
(1,M,L,T,J);

%  Fit the MLMM
[lk,iteration,G,Ng,J,T,L,M,PH_g,PXgivenH,TauGivenH,CondResProd,Est_h,np
,AIC,BIC]=...
    MLMM(Data4D,Data2D,G,Ng,L,M,J,T,PH_g,PXgivenH,TauGivenH,CondResProd)

%% === 3-clutsers 3-Classes MLMM ===
% Specify the parameters:
G = 31; Ng= 50; J=10;  T=2;
L = 3;  % Number of latent clusters
M = 3;  % Number of latent classes

% generate the starting values
[PH_g,PXgivenH,TauGivenH,TauGivenHT,CondResProd]=StartingValue_RandomV1
(1,M,L,T,J);

%  Fit the MLMM
[lk,iteration,G,Ng,J,T,L,M,PH_g,PXgivenH,TauGivenH,CondResProd,Est_h,np
,AIC,BIC]=...
    MLMM(Data4D,Data2D,G,Ng,L,M,J,T,PH_g,PXgivenH,TauGivenH,CondResProd)
```

Appendix B2

Fitting MLCM using the MDLV MATLAB toolbox (cf. Table 2, Table 3, and Table 4)

The script for fitting MLCM is provided below in '`EVSdata_MLCM.m`'. The data `ItemsT1_3D` and `ItemsT2_3D` contain responses to the 10 items for Wave 3 and Wave 4, respectively. Each data set is a matrix of $G \times Ng \times J$. Four models are fitted to each data set: 2 clusters by 2 classes, 2 clusters by 3 classes, 3 clusters by 2 classes, and 3 clusters by 3 classes. After specifying the model parameters, the starting values can be generated using the function '`StartingValue_RandomV1`'. The parameters of MLCM are estimated using the function '`MLCM.m`' which requires the sub-function '`Loglikelihood_MLCM.m`' to calculate the log-likelihood value.

`EVSdata_MLCM.m`:

```
%% ======   Fit MLCM   ======
%% === Wave 3 ===
%% W3 === 2-clutsers 2-Classes MLCM ===
% Specify the parameters:
G = 31; % Number of groups
Ng= 50; % Number of subjects per group
```

```
J=10;    % Number of items
T=1;     % Number of time points
L = 2;   % Number of latent clusters
M = 2;   % Number of latent classes
% generate the starting values
[PH_g,PXgivenH,TauGivenH,TauGivenHT,CondResProd]=StartingValue_RandomV1
(1,M,L,T,J);

[lk,iteration,G,n_g,J,L,M,est_PH_g,est_PXgivenH,est_CondResProd,Est_h,n
p,AIC,BIC]=...
    MLCM(ItemsT1_3D,G,Ng,L,M,J,PH_g,PXgivenH,CondResProd)
%% W3 === 2-clutsers 3-Classes MLCM ===
% Specify the parameters:
G = 31; Ng= 50; J=10; T=1;
L = 2;   % Number of latent clusters
M = 3;   % Number of latent classes

% generate the starting values
[PH_g,PXgivenH,TauGivenH,TauGivenHT,CondResProd]=StartingValue_RandomV1
(1,M,L,T,J);

[lk,iteration,G,n_g,J,L,M,est_PH_g,est_PXgivenH,est_CondResProd,Est_h,n
p,AIC,BIC]=...
    MLCM(ItemsT1_3D,G,Ng,L,M,J,PH_g,PXgivenH,CondResProd)

%% W3 === 3-clutsers 2-Classes MLCM ===
% Specify the parameters:
G = 31; Ng= 50; J=10; T=1;
L = 3;   % Number of latent clusters
M = 2;   % Number of latent classes

% generate the starting values
[PH_g,PXgivenH,TauGivenH,TauGivenHT,CondResProd]=StartingValue_RandomV1
(1,M,L,T,J);

[lk,iteration,G,n_g,J,L,M,est_PH_g,est_PXgivenH,est_CondResProd,Est_h,n
p,AIC,BIC]=...
    MLCM(ItemsT1_3D,G,Ng,L,M,J,PH_g,PXgivenH,CondResProd)

%% W3 === 3-clutsers 3-Classes MLCM ===
% Specify the parameters:
G = 31; Ng= 50; J=10; T=1;
L = 3;   % Number of latent clusters
M = 3;   % Number of latent classes

% generate the starting values
[PH_g,PXgivenH,TauGivenH,TauGivenHT,CondResProd]=StartingValue_RandomV1
(1,M,L,T,J);

[lk,iteration,G,n_g,J,L,M,est_PH_g,est_PXgivenH,est_CondResProd,Est_h,n
p,AIC,BIC]=...
    MLCM(ItemsT1_3D,G,Ng,L,M,J,PH_g,PXgivenH,CondResProd)
%% === Wave 4 ===
%% W4 === 2-clutsers 2-Classes MLCM ===
% Specify the parameters:
```

```
G = 31; Ng= 50; J=10; T=1;
L = 2;  % Number of latent clusters
M = 2;  % Number of latent classes
% generate the starting values
[PH_g,PXgivenH,TauGivenH,TauGivenHT,CondResProd]=StartingValue_RandomV1
(1,M,L,T,J);

[lk,iteration,G,n_g,J,L,M,est_PH_g,est_PXgivenH,est_CondResProd,Est_h,n
p,AIC,BIC]=...
    MLCM(ItemsT2_3D,G,Ng,L,M,J,PH_g,PXgivenH,CondResProd)

%% W4 === 2-clutsers 3-Classes MLCM ===
% Specify the parameters:
G = 31; Ng= 50; J=10; T=1;
L = 2;  % Number of latent clusters
M = 3;  % Number of latent classes
% generate the starting values
[PH_g,PXgivenH,TauGivenH,TauGivenHT,CondResProd]=StartingValue_RandomV1
(1,M,L,T,J);

[lk,iteration,G,n_g,J,L,M,est_PH_g,est_PXgivenH,est_CondResProd,Est_h,n
p,AIC,BIC]=...
    MLCM(ItemsT2_3D,G,Ng,L,M,J,PH_g,PXgivenH,CondResProd)
%% W4 === 3-clutsers 2-Classes MLCM ===
% Specify the parameters:
G = 31; Ng= 50; J=10; T=1;
L = 3;  % Number of latent clusters
M = 2;  % Number of latent classes

% generate the starting values
[PH_g,PXgivenH,TauGivenH,TauGivenHT,CondResProd]=StartingValue_RandomV1
(1,M,L,T,J);

[lk,iteration,G,n_g,J,L,M,est_PH_g,est_PXgivenH,est_CondResProd,Est_h,n
p,AIC,BIC]=...
    MLCM(ItemsT2_3D,G,Ng,L,M,J,PH_g,PXgivenH,CondResProd)
%% W4 === 3-clutsers 3-Classes MLCM ===
% Specify the parameters:
G = 31; Ng= 50; J=10; T=1;
L = 3;  % Number of latent clusters
M = 3;  % Number of latent classes
% generate the starting values
[PH_g,PXgivenH,TauGivenH,TauGivenHT,CondResProd]=StartingValue_RandomV1
(1,M,L,T,J);

[lk,iteration,G,n_g,J,L,M,est_PH_g,est_PXgivenH,est_CondResProd,Est_h,n
p,AIC,BIC]=...
    MLCM(ItemsT2_3D,G,Ng,L,M,J,PH_g,PXgivenH,CondResProd)
```

Appendix B3

Fitting LMM using the MDLV MATLAB toolbox (cf. Table 2 and Table 5)

The script for fitting LMM is included in 'EVSdata_LMM.m' as shown below. Data3D is created with the script 'EVSdata.m' (see Appendix A2). The starting values for model parameters are required (Start_PX, Start_Tau, and Start_CondResProd) and are randomly generated in this example. Four models with number of latent classes ranging from 2 to 5 are fitted. The number of latent classes is specified by assigning a value to M. The main function to estimate the parameters of the LMM is 'LMM.m'. The sub-function 'Loglikelihood_LMM.m' (nested in 'LMM.m') is used to calculate the log-likelihood value.

EVSdata_LMM.m:

```
%% ======   Fit LMM   ======
%% === 2-Classes LMM ===
N=(31*50);    % Number of Total subjects (G*Ng)
T=2;          % Number of time points
J=10;         % Number of items
M=2;          % Number of latent classes
% generate the starting values: PX
Start_PX=rand(M,1); Start_PX=Start_PX/csum(Start_PX);

% Starting value for Tau (random start)
Start_Tau = (0.1*rand(M,M)+ones(M,M))/M;
Start_Tau = rdiv(Start_Tau,rsum(Start_Tau));
Start_Tau=reshape(repmat(Start_Tau,1,T),M,M,T);

%Start_CondResProd;
Start_CondResProd =rand(M,J);

[lk,iteration,n,T,J,M,pi1,TranMatrix,rho,np,AIC,BIC]...
    = LMM(Data3D,Start_PX,Start_Tau,Start_CondResProd,N,T,J,M)

%% === 3-Classes LMM ===
N=(31*50);    T=2;    J=10;
M=3;          % Number of latent classes

Start_PX=rand(M,1); Start_PX=Start_PX/csum(Start_PX);
Start_Tau = (0.1*rand(M,M)+ones(M,M))/M;
Start_Tau = rdiv(Start_Tau,rsum(Start_Tau));
Start_Tau=reshape(repmat(Start_Tau,1,T),M,M,T);
Start_CondResProd =rand(M,J);

[lk,iteration,n,T,J,M,pi1,TranMatrix,rho,np,AIC,BIC]...
    = LMM(Data3D,Start_PX,Start_Tau,Start_CondResProd,N,T,J,M)

%% === 4-Classes LMM ===
N=(31*50);    T=2;    J=10;
M=4;          % Number of latent classes

Start_PX=rand(M,1); Start_PX=Start_PX/csum(Start_PX);
Start_Tau = (0.1*rand(M,M)+ones(M,M))/M;
Start_Tau = rdiv(Start_Tau,rsum(Start_Tau));
Start_Tau=reshape(repmat(Start_Tau,1,T),M,M,T);
```

```
Start_CondResProd =rand(M,J);

[lk,iteration,n,T,J,M,pi1,TranMatrix,rho,np,AIC,BIC]...
    = LMM(Data3D,Start_PX,Start_Tau,Start_CondResProd,N,T,J,M)

%% === 5-Classes LMM ===
N=(31*50);    T=2;    J=10;
M=5;          % Number of latent classes

Start_PX=rand(M,1); Start_PX=Start_PX/csum(Start_PX);
Start_Tau = (0.1*rand(M,M)+ones(M,M))/M;
Start_Tau = rdiv(Start_Tau,rsum(Start_Tau));
Start_Tau=reshape(repmat(Start_Tau,1,T),M,M,T);
Start_CondResProd =rand(M,J);

[lk,iteration,n,T,J,M,pi1,TranMatrix,rho,np,AIC,BIC]...
    = LMM(Data3D,Start_PX,Start_Tau,Start_CondResProd,N,T,J,M)
```

Appendix B4

Fitting LCM using the MDLV MATLAB toolbox (cf. Table 2 and Table 5)

The script for fitting LCM to the sample data from EVS is given below as "EVSdata_LCM.m". Data ItemsT1_2D and ItemsT2_2D can be obtained by running the script 'EVSdata.m' provided in Appendix A2. Four models, ranging from 2 to 5 latent classes, are fitted to the data for Wave 3 and Wave 4 separately. The function 'aggregate.m' is used to construct a matrix of unique response patterns (for the 10 items) and a vector of the corresponding frequencies for the patterns. The main function 'LCM.m' is used to estimate the model parameters.

EVSdata_LCM.m:

```
%% ======   Fit LCM   ======
% [lk,it,n,J,pi1,Rho,np,AIC,BIC] = LCM_V2(S,yv,M,start)
% start: 1 -> random starts
% Data LCM:  Wave3:ItemsT1_2D
%            Wave4:ItemsT2_2D
[S_T1,yv_T1] = aggregate(ItemsT1_2D);
[S_T2,yv_T2] = aggregate(ItemsT2_2D);
% aggregate function: obtain matrix with unique response patterns and
% corresponding frequencies for each pattern.

%% === Wave 3 ===
% Specify the parameters:
M = 2;  % Number of latent classes = 2
[lk,it,n,J,pi1,Rho,np,AIC,BIC]=LCM(S_T1,yv_T1,M,1)
M = 3;  % Number of latent classes = 3
[lk,it,n,J,pi1,Rho,np,AIC,BIC]=LCM(S_T1,yv_T1,M,1)
M = 4;  % Number of latent classes = 4
[lk,it,n,J,pi1,Rho,np,AIC,BIC]=LCM(S_T1,yv_T1,M,1)
M = 5;  % Number of latent classes = 5
```

```
[lk,it,n,J,pi1,Rho,np,AIC,BIC]=LCM(S_T1,yv_T1,M,1)

%% === Wave 4 ===
% Specify the parameters:
M = 2;  % Number of latent classes = 2
[lk,it,n,J,pi1,Rho,np,AIC,BIC]=LCM(S_T2,yv_T2,M,1)
M = 3;  % Number of latent classes = 3
[lk,it,n,J,pi1,Rho,np,AIC,BIC]=LCM(S_T2,yv_T2,M,1)
M = 4;  % Number of latent classes = 4
[lk,it,n,J,pi1,Rho,np,AIC,BIC]=LCM(S_T2,yv_T2,M,1)
M = 5;  % Number of latent classes = 5
[lk,it,n,J,pi1,Rho,np,AIC,BIC]=LCM(S_T2,yv_T2,M,1)
```

Appendix C

The MDLV MATLAB toolbox

The data formats for fitting LCM, LMM, MLCM, and MLMM with the MDLV MATLAB toolbox are described in Appendix C1. The inputs and outputs of the main estimation functions for each model are annotated in Appendix C2. In addition, scripts for simulating data according to each of the four types of models are provided in Appendix C3. Appendix C3 also gives detailed descriptions of several utility functions used in the toolbox. All of these functions are organized in the "MDLV_MATLAB" folder and are available by request.

Appendix C1: Data Format

- *LCM*.
  The data required for LCM consist of a rectangular matrix (S) with rows being unique response patterns in the data and a vector of frequencies for the response patterns (yv). For data matrices with rows corresponding to response vector of a single subject, the utility function 'aggregate.m' can be used to form the matrix of unique response patterns and the vector of corresponding frequencies.

- *LMM*.
  The data set for LMM is a three-dimensional matrix: individuals by items by time points ($N \times J \times T$). The '2Dto3D.m' function can be used to transform a rectangular matrix of individuals by items and time points to a three-dimensional data matrix needed when fitting LMM.

- *MLCM*.
  The data set for MLCM is a three-dimensional matrix of ($G \times Ng \times J$), where G is the number of groups, Ng is the number of individuals in group g , and J is the number of items.

- *MLMM*.
  The data set for MLMM is a four-dimensional matrix of ($G \times Ng \times J \times T$), where G is the number of groups, Ng is the number of individuals in group g , J denotes the number of the items, and T is the number of occasions.

Appendix C2: Inputs and outputs of main estimation functions

The main estimation functions for each of the four models for discrete latent variables are LCM.m, LMM.m, MLCM.m, and MLMM.m. The input and output variables for each model are listed below with corresponding descriptions.

LCM.m

```
function [lk,it,n,J,pi1,Rho,np,AIC,BIC] = LCM(S,yv,M,start);
```

Input:

Parameters		Dimension
S	Matrix of unique response patterns for all items	#unique vector X J
Yv	Vector of corresponding frequencies of the unique response patterns in matrix *S*	#unique vector X 1
M	Number of latent classes	Scalar
Start	Specify the type of starting values: 0 = deterministic (estimated from observed data), 1 = random, and 2 = load user specified values from the file 'start.mat'.	Depending on number of classes and number of items

Output:

Parameters		Dimension
lk	Log-likelihood value	Scalar
it	Number of iterations	Scalar
n	Number of subjects	Scalar
J	Number of items	Scalar
pi1	Estimated latent class probabilities	$M \times 1$
Rho	Estimated conditional response probability	$M \times J$
np	Number of parameters	Scalar
AIC	Akaike information criterion	Scalar
BIC	Bayesian information criterion	Scalar

LMM.m

```
function [lk,iteration,n,T,J,M,pi1,TranMatrix,rho,np,AIC,BIC] =
LMM(X,pi1,tau,rho,n,T,J,M)
```

Input:

Parameters		Dimension
X	Data Matrix	$N \times J \times T$
pi1	Starting latent class probabilities	$M \times 1$
tau	Starting latent transition matrix	$M \times M$
rho	Starting conditional response probabilities	$M \times J$
n	Number of subjects	Scalar
T	Number of occasions	Scalar
J	Number of items	Scalar
M	Number of latent classes	Scalar

Output:

Parameters		Dimension
lk	Log-likelihood value	Scalar
iteration	Number of iterations	Scalar
n	Number of subjects	Scalar
T	Number of time points	Scalar
J	Number of items	Scalar
M	Number of latent classes	Scalar
pi1	Estimated latent classes probabilities	$M \times 1$
TranMatrix	Estimated latent transition matrix	$M \times M$
rho	Estimated conditional response probabilities	$M \times J$
np	Number of parameters	Scalar
AIC	Akaike information criterion	Scalar
BIC	Bayesian information criterion	Scalar

MLCM.m

```
function
[lk,iteration,G,n_g,J,L,M,est_PH_g,est_PXgivenH,est_CondResProd,Est_h,n
p,AIC,BIC]=...
    MLCM(Data3D,G,n_g,L,M,J,est_PH_g,est_PXgivenH,est_CondResProd)
```

Input:

Parameters		Dimension
Data3D	Data Matrix	$G \times N_g \times J$
G	Number of groups	Scalar
n_g	Number of subjects per group	Scalar
L	Numbers of latent clusters	Scalar
M	Number of latent classes	Scalar
J	Number of items	Scalar
est_PH_g	Starting latent class probabilities	$M \times 1$
est_PXgivenH	Starting latent transition matrix	$M \times M$
est_CondResProd	Starting conditional response probabilities	$M \times J$

Output:

Parameters		Dimension
lk	Log-likelihood value	Scalar
iteration	Number of iterations	Scalar
G	Number of groups	Scalar
n_g	Number of subjects per group	Scalar
J	Number of items	Scalar
L	Number of latent clusters	Scalar
M	Number of latent classes	Scalar
est_PH_g	Estimated latent cluster probabilities	$L \times 1$
est_PXgivenH	Estimated conditional latent class probabilities	$M \times L$
est_CondResProd	Estimated conditional response probabilities	$M \times J$
Est_h	Estimated latent cluster membership	$G \times 1$
np	Number of parameters	Scalar

AIC	Akaike information criterion	Scalar
BIC	Bayesian information criterion	Scalar

MLMM.m

```
function
[lk,iteration,G,Ng,J,T,L,M,PH_g,PXgivenH,TauGivenH,CondResProd,Est_h,np
,AIC,BIC]=...
    MLMM(Data4D,Data2D,G,Ng,L,M,J,T,PH_g,PXgivenH,TauGivenH,CondResProd)
```

Input:

Parameters		Dimension
Data4D	Data Matrix	$G \times Ng \times J \times T$
Data2D	Data Matrix	$(G \times Ng \times T) \times J$
G	Number of groups	Scalar
Ng	Number of subjects per group	Scalar
L	Number of latent clusters	Scalar
M	Number of latent classes	Scalar
J	Number of items	Scalar
T	Number of time points	Scalar
PH_g	Starting latent class probabilities	$M \times 1$
PXgivenH	Starting latent transition matrix	$M \times M \times L$
CondResProd	Starting conditional response probabilities	$M \times J$

Output:

Parameters		Dimension
lk	Log-likelihood value	Scalar
iteration	Number of iteration	Scalar
G	Number of groups	Scalar
n_g	Number of subjects per group	Scalar
J	Number of items	Scalar
T	Number of time points	Scalar
L	Number of latent clusters	Scalar
M	Number of latent classes	Scalar
PH_g	Estimated latent cluster probabilities	$L \times 1$
PXgivenH	Estimated conditional latent class probabilities	$M \times L$
TauGivenH	Estimated conditional latent transition matrix	L of $M \times M$
CondResProd	Estimated conditional response probabilities	$M \times J$
Est_h	Estimated latent cluster membership	$G \times 1$
np	Number of parameters	Scalar
AIC	Akaike information criterion	Scalar
BIC	Bayesian information criterion	Scalar

Appendix C3: Data Simulation Scripts and Utility Functions

The MDLV MATLAB toolbox provides functions for simulating data from the specified model using given parameters and supporting utility functions.

Data Simulation

- Simulate_LCM.m: A script that can be used to simulate data based on a LCM with the specified number of latent classes, latent class probabilities, and conditional response probabilities. The output is an individuals by items ($N \times J$) data matrix.
- Simulate_LMM.m: A script that can be used to simulate data based on a LMM. The function requires: (1) the number of latent classes, (2) the number of time points, (3) a vector of latent class probabilities, (4) a transition matrix, and (5) the conditional response probabilities. The output is a ($N \times J \times T$) data matrix.
- Simulate_MLCM.m: A script that can be used to simulate data based on a MLCM. The number of individuals per group is fixed to be the same (N_g).

  Groups are first assigned to one of the latent clusters, and individuals of each group are then assigned to latent classes based on the latent class probabilities given their group's latent cluster membership. The responses of each individual are simulated based on his/her latent class membership. This function generates a three-dimensional data matrix of $G \times N_g \times J$.
- Simulate_MLMM.m: A script that can be used to simulate data based on a MLMM. In addition to specifications as required in 'Simulate_MLCM.m', this function also needs the conditional transition matrix for each latent cluster. The latent class memberships at second and subsequent time points are generated from the transitional probabilities. All other steps for simulating the data follow the same procedure as explained in 'Simulate_MLCM.m'. The output of this function is a four-dimensional ($G \times N_g \times J \times T$) data matrix.

Utility Functions

- aggregate.m: Transforms an individual by item matrix into a matrix containing unique response patterns in the rows, and a vector of corresponding frequencies for the unique patterns.
- rsum.m: Sum over each row of a vector or matrix.
- csum.m: Sum over each column of a matrix.
- diagv.m: Computes $N = \text{diag}(v) \times M$ that avoids the *diag* operator.
- Sample_discrete.m: Sample from a non-uniform discrete distribution. Syntax of sample discrete ([0.3 0.7], 1, 5) generates a row vector of 5 random integers from {1,2}, where the probability of being 1 is 0.3 and the probability of being 2 is 0.7.
- StartingValue_RandomV1.m: Generate random starting values for model parameters in a MLCM and a MLMM.