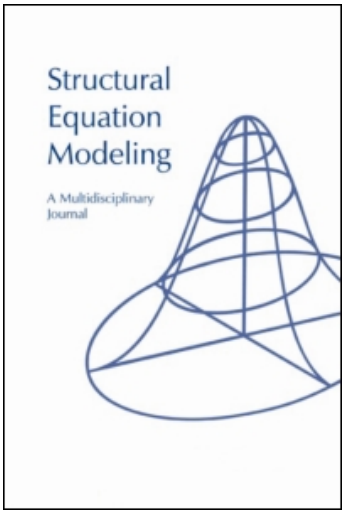
This article was downloaded by: *[University of California, Los Angeles]*
On: *9 February 2011*
Access details: *Access Details: [subscription number 918974530]*
Publisher *Psychology Press*
Informa Ltd Registered in England and Wales Registered Number: 1072954 Registered office: Mortimer House, 37-41 Mortimer Street, London W1T 3JH, UK



Structural Equation Modeling: A Multidisciplinary Journal
Publication details, including instructions for authors and subscription information:
http://www.informaworld.com/smpp/title~content=t775653699

Addressing the Problem of Switched Class Labels in Latent Variable Mixture Model Simulation Studies

Stephen J. Tueller[a]; Scott Drotar[a]; Gitta H. Lubke[a]
[a] University of Notre Dame,

Online publication date: 07 January 2011

To cite this Article Tueller, Stephen J. , Drotar, Scott and Lubke, Gitta H.(2011) 'Addressing the Problem of Switched Class Labels in Latent Variable Mixture Model Simulation Studies', Structural Equation Modeling: A Multidisciplinary Journal, 18: 1, 110 — 131
To link to this Article: DOI: 10.1080/10705511.2011.534695
URL: http://dx.doi.org/10.1080/10705511.2011.534695

PLEASE SCROLL DOWN FOR ARTICLE

Full terms and conditions of use: http://www.informaworld.com/terms-and-conditions-of-access.pdf

This article may be used for research, teaching and private study purposes. Any substantial or systematic reproduction, re-distribution, re-selling, loan or sub-licensing, systematic supply or distribution in any form to anyone is expressly forbidden.

The publisher does not give any warranty express or implied or make any representation that the contents will be complete or accurate or up to date. The accuracy of any instructions, formulae and drug doses should be independently verified with primary sources. The publisher shall not be liable for any loss, actions, claims, proceedings, demand or costs or damages whatsoever or howsoever caused arising directly or indirectly in connection with or arising out of the use of this material.

Structural Equation Modeling, 18:110–131, 2011
Copyright © Taylor & Francis Group, LLC
ISSN: 1070-5511 print/1532-8007 online
DOI: 10.1080/10705511.2011.534695

Psychology Press
Taylor & Francis Group

Downloaded By: [University of California, Los Angeles] At: 20:58 9 February 2011

TEACHER'S CORNER

Addressing the Problem of Switched Class Labels in Latent Variable Mixture Model Simulation Studies

Stephen J. Tueller, Scott Drotar, and Gitta H. Lubke
University of Notre Dame

The discrimination between alternative models and the detection of latent classes in the context of latent variable mixture modeling depends on sample size, class separation, and other aspects that are related to power. Prior to a mixture analysis it is useful to investigate model performance in a simulation study that reflects the research settings. Multiple data sets are generated under 1 or more models, and alternative models are fitted to the data. The aggregation of results over multiple data sets is complicated by the fact that mixture models are only identified up to a permutation of the class labels. Estimated class labels are arbitrary, with the effect that the estimated parameters for Class 1 could be incorrectly labeled as Class 2, Class 3, and so forth, relative to their data generating labels. In a simulation study, the detection of switched labels needs to be automated. Switched class labels are not necessarily simple to detect. This article describes different possible scenarios of switched class labels, and develops an algorithm implemented in R that (a) detects switched labels, and (b) provides information that can be used to either correct class labels or to discard a particular data set from a simulation if class labels are ambivalent. The algorithm is useful in Monte Carlo simulations involving latent variable mixture models.

Latent variable mixture modeling has become popular for the analysis of data in a variety of areas within the behavioral sciences (e.g., B. O. Muthén & Asparouhov, 2006; Neale, Aggen, Maes, Kubarych, & Schmitt, 2006; Nylund, Bellmore, Nishina, & Graham, 2007; Torppa et al., 2007). The popularity is due mainly to the fact that latent variable mixture modeling permits the investigation of differences between groups without having to rely on observed data to form the groups. The study of latent groups or classes can be carried out longitudinally or

Correspondence should be addressed to Gitta H. Lubke, Department of Psychology, 118 Haggar Hall, University of Notre Dame, Notre Dame, IN 46556, USA. E-mail: glubke@nd.edu

at single time points, and increasingly complex mixture models have been proposed (e.g., Jedidi, Jagpal, & DeSarbo, 1997; B. O. Muthén, 2004; Muthén, Asparouhov, & Rebollo, 2006; Vermunt, 2003). A typical empirical mixture analysis consists of fitting and comparing several alternative models with different within-class structures, increasing numbers of classes, or both. It has been shown that the detection of classes, and, more generally, the discrimination between alternative models, is a matter of power. Just as in comparisons of observed groups, sample size and the size of the mean differences between classes have an impact on the detection of the differences, and hence on which model is selected as the best fitting model (Lubke & Neale, 2006, 2008). Simulation studies can be conducted to assess model performance and selection in preparation for or in conjunction with empirical latent variable mixture modeling analyses.

Aggregating parameter estimates across the multiple simulated data sets is complicated by the fact that class labels are arbitrary in mixture models. Mixture models are only identified up to permutations of the class labels (McLachlan & Peel, 2000; Titterington, Smith, & Makov, 1985). For example, the data generating values for the factor variances in Classes 1 and 2 might be 2 and 4. The estimates from fitting a latent variable mixture model (LVMM) to a given data set may be 3.9 for Class 1 and 2.1 for Class 2. Here the labels are clearly switched. Because class labels are potentially switched across data sets, aggregating parameter estimates over potentially mislabeled classes is undesirable. This is not a problem for fit indexes, because these are unaffected by switched labels.

Switching can be prevented to some extent by providing true parameter values as starting values. However, to realistically mimic an empirical study, random starting values should be provided rather than the true parameter values that were used to generate the data. Realistically mimicking empirical studies also precludes imposing model constraints beyond those needed for model identification to potentially prevent switching (e.g., Diebolt & Robert, 1994). In addition, not all choices of constraints are guaranteed to eliminate switching, and different choices might yield different permutations of the class labels. Because the use of this type of constraint is confined to parameters that are ordered across classes, their use is restricted to a subset of simulation conditions. In addition, these constraints are typically expressed as linear or nonlinear inequality constraints, which might be problematic to implement using existing model fitting software.

An alternative to the prevention of label switching is to inspect parameter estimates after estimation has completed. Continuing the preceding example, if the data generating values for the factor variances in Classes 1 and 2 were 2 and 4, but estimates for a given data set are 3.9 and 2.1, we would conclude that the class labels are switched. However, as the number of Monte Carlo replications in a simulation study is usually large, it is not feasible to check every data set by hand. In addition, parameter inspection methods are not guaranteed to lead to unique switched label detection. As a hypothetical example using a structural equation mixture model (SEMM) with two factors and four classes, inspecting the factor variances might suggest new labels of 3, 2, 1, 4, whereas inspecting the regression parameters between the two factors might suggest new labels of 2, 3, 1, 4.

In summary, the problem of switched class labels needs to be addressed without relying on starting values, model constraints, or the inspection of parameter estimates. This work describes an algorithm that automatically detects switched class labels and provides information that can be used to correctly relabel class parameters relative to the data generating labels in a simulation study.

Downloaded By: [University of California, Los Angeles] At: 20:58 9 February 2011

The article is organized as follows. First, we describe the general LVMM, including constraints needed for LVMM identification. Second, the problem of switched class labels is described in detail. Third, an approach to post-hoc detection of switched class labels is presented (rather than *a priori* prevention). This approach is based on information in the class assignment matrix and is formalized in a switched label detection algorithm. Fourth, an implementation of the switched label detection algorithm in R is described and illustrated using output produced by M*plus* (L. K. Muthén & Muthén, 2007), the most widely used software for fitting LVMMs to data. Finally, methods for automating the correction of switched parameter labels are described.

Downloaded By: [University of California, Los Angeles] At: 20:58 9 February 2011

THE LATENT VARIABLE MIXTURE MODEL

The distribution of the observed data in a mixture model is assumed to be a mixture distribution. Mixture distributions are a weighted sum of two or more component distributions, and the weights correspond to the relative size of a component. Details of general finite mixture models can be found in Everitt and Hand (1981), Titterington et al. (1985), or McLachlan and Peel (2000).

The LVMM is a special case of the general finite mixture model, and assumes multivariate normal component distributions. LVMMs for K classes are fitted using a mixture distribution with K components. The LVMM is given by

$$f(\boldsymbol{y}) = \sum_{k=1}^{K} \pi_k \phi_k (\boldsymbol{y}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{1}$$

where $\boldsymbol{y}$ is a vector of p observed continuous variables, K is the number of classes, π_k are the class proportions, $\sum_{k=1}^{K} \pi_k = 1$, and ϕ_k are multivariate normal probability density functions with class-specific mean vectors $\boldsymbol{\mu}_k$ and class-specific covariance matrices $\boldsymbol{\Sigma}_k$.

LVMM submodels are specified by imposing a particular structure on the mean vectors and covariance matrices of the K component distributions. Imposing a common factor model (CFA; e.g., Jöreskog, 1966) leads to the factor mixture model (FMM; Arminger, Stein, & Wittenberg, 1999; B. O. Muthén & Shedden, 1999; Yung, 1997). Imposing an item-response model (IRT; e.g., Lord, 1953) leads to the mixture IRT model (e.g., B. O. Muthén & Asparouhov, 2006; Rost, 1988). Imposing latent growth curve models (LGC; e.g., Meredith & Tisak, 1990) leads to the growth mixture model (GMM; e.g., B. O. Muthén & Shedden, 1999). Imposing a structural equation model (SEM; e.g., Bollen, 1989) on each component leads to the SEMM (Dolan & van der Maas, 1998; Jedidi et al., 1997). In this work, the FMM will be used as an illustration. The FMM was selected because of its relationship to the other LVMMs mentioned herein: The two-parameter IRT model is equivalent to the CFA model with binary indicators, where parameters in one model are transformations of parameters in the other (B. O. Muthén & Asparouhov, 2006). The LGC model is a special case of the CFA where parameters are fixed to estimate a desired growth trajectory (Meredith & Tisak, 1990). The SEM generalizes the multidimensional CFA to incorporate relationships among latent factors (Bollen, 1989).

The K multivariate normal components of a K-class FMM are structured as

$$\boldsymbol{\mu}_k = \boldsymbol{\nu}_k + \boldsymbol{\Lambda}_k \boldsymbol{\alpha}_k \tag{2}$$

$$\boldsymbol{\Sigma}_k = \boldsymbol{\Lambda}_k \boldsymbol{\Psi}_k \boldsymbol{\Lambda}_k' + \boldsymbol{\Theta}_k, \tag{3}$$

where $\boldsymbol{\nu}_k$ is a $p \times 1$ vector of equation intercepts in class k, p is the number of observed variables, $\boldsymbol{\Lambda}_k$ is a $p \times m_k$ matrix of factor loadings in class k, m_k is the number of factors in class k, $\boldsymbol{\alpha}_k$ is an $m_k \times 1$ vector of factor means in class k, $\boldsymbol{\Psi}_k$ is the $m_k \times m_k$ covariance matrix for the factors in class k, and $\boldsymbol{\Theta}_k$ is a $p \times p$ covariance matrix of the measurement errors with error variances on the diagonal. The model-implied within-class mean vector and covariance matrix given in Equations 2 and 3 permit specification of factor means, factor variances, factor covariances, factor loadings, intercepts, and error variances. However, not all of these parameters are identified, hence estimating the FMM is subject to identifiability constraints.

LVMM Identification

For an LVMM to be identified, the within-class models need to be identified. The required identifiability constraints are the same as those in multiple-group latent variable models. For instance, the scale of the latent factors needs to be fixed, either by linking the scale to an observed item (e.g., fix the corresponding loading), or by fixing the factor variance. Because multiple latent groups are investigated, also the mean of the factor needs to be fixed in one group, thereby permitting the estimation of factor mean differences in the other groups (Jöreskog, 1971; B. O. Muthén & Asparouhov, 2002; Sörbom, 1974). Typically, the factor mean(s) in one class are set to zero, making this class the reference class. As explained in the introduction, model constraints, including required identifiability constraints, do not guarantee the prevention of switched class labels. The label switching problem is described in detail next.

THE CLASS LABEL SWITCHING PROBLEM

Even if minimum identifiability constraints are satisfied for an LVMM, the model is only identified up to a permutation of the class labels. Conceptually, this means that the labels of the estimated parameters might not match the data generating labels for simulated data. This is called the label switching problem in mixture model literature. There are $K!$ possible permutations of the K class labels, where ! is the factorial operator. For example, a three-class LVMM can have the classes labeled as $\{1, 2, 3\}$, $\{1, 3, 2\}$, $\{2, 3, 1\}$, $\{2, 1, 3\}$, $\{3, 2, 1\}$, or $\{3, 1, 2\}$ without changing the model fit.

The consequences of switched labels are illustrated on four data sets in which a one-factor, two-class mixture model is fit to the data. If switched labels are ignored, estimates of the factor variance in Class 1 (Ψ_1) and the factor variance in Class 2 (Ψ_2) might look like those in the left side of Table 1. Incorrect summary statistics are shown in the lower left margin of Table 1. The parameter labels for factor variances can be corrected, as is illustrated on the right side of Table 1. Correct summary statistics can then be computed, as shown in the lower right margin

Downloaded By: [University of California, Los Angeles] At: 20:58 9 February 2011

TABLE 1
Example Simulation Results with Switched and Corrected Labels

Switched	Ψ_1 *(1)*	Ψ_2 *(3)*	*Corrected*	Ψ_1 *(1)*	Ψ_2 *(3)*
data1	3.1	0.9		0.9	3.1
data2	1.1	3.2		1.1	3.2
data3	2.8	1.2		1.2	2.8
data4	2.9	0.8		0.8	2.9
M	2.475	1.525		1.0	3.0

Note. Ψ_1 and Ψ_2 are the factor variances in Classes 1 and 2, respectively, with population values in parentheses.

Downloaded By: [University of California, Los Angeles] At: 20:58 9 February 2011

of Table 1. Although visual inspection works in this small example, it will not work in general, as described in the introduction. In the next section we show how the information in the class assignment matrix can be used to detect switched labels and to provide information that can be used to correct switched class labels.

CLASS ASSIGNMENT APPROACH TO DETECTING SWITCHED LABELS

This section introduces an algorithm that can (a) detect switched class labels, and (b) provide information that can be used to correct parameter labels. In later sections, this algorithm is implemented in the `LBLinfo.r` script for the freely available software `R` (R Development Core Team, 2008). The script can be used with LVMM output from any software program that can save assigned class membership to a file. Because M*plus* (L. K. Muthén & Muthén, 2007) is the most widely used software for fitting LVMMs to data, use of the R script will be documented herein using M*plus* output.

In the sections that follow, the switched label detection algorithm is developed and illustrated. First, we describe the class assignment matrix. Second, the algorithm to detect switched labels is developed. The algorithm uses the class assignment matrix as input, and returns correct class labels as output. Third, conditions under which the algorithm will work are described, limitations of the algorithm are noted, and criteria for ensuring these conditions are met are described. To prevent the spurious detection of switched labels, the `LBLinfo.r` script includes several checks to evaluate whether these criteria are met. Fourth, we illustrate how to use the algorithm as implemented in the `LBLinfo.r` script.

The Class Assignment Matrix

When fitting an LVMM to data, each participant can be assigned to the latent class to which he or she most likely belongs. In simulation studies, true class membership is known and can be compared to assigned class membership. For each data set, a simple way to summarize assigned and true class membership is using the class assignment matrix. Consider the simple example given in Table 2. Here, a data set with 10 participants is generated where the data generating label of six participants is Class 1 and for the other four participants the label is Class 2. After a hypothetical LVMM is fit to these data, four of the Class 1 participants are correctly assigned

TABLE 2
Example True Class Labels, Assigned Class Labels, and Corresponding Class Assignment Matrix

True	1	2	1	1	2	1	1	2	2	1
Assigned	1	2	2	1	1	2	1	2	2	1

	True 1	*True 2*
Assign 1	4	1
Assign 2	2	3

Note. The upper table represents raw data, and the lower table is the class assignment matrix obtained by cross-tabulating the two rows of the upper table. In R, the `table()` function can be used to obtain the lower table from the upper table.

Downloaded By: [University of California, Los Angeles] At: 20:58 9 February 2011

to Class 1, and three of the Class 2 participants are correctly assigned to Class 2 as can be seen on the diagonal of the class assignment matrix in the lower portion of Table 2.

In general, a class assignment matrix M is a $K \times K$ cross-tabulation matrix. Herein we adopt the convention that true class labels are in the columns and that assigned class labels are in the rows, as illustrated in Table 2. Note that because column labels are the data generating class labels, column labels are fixed, and only row labels are subject to potential switching. This is illustrated in Table 3. It is quite easy to see in Table 3 that merely changing Assign 1 to Assign 2 and vice versa corrects the class assignment matrix. It is this simple idea that is the core of the switched class label detection algorithm.

A CLASS ASSIGNMENT BASED ALGORITHM TO DETECT SWITCHED LABELS

If the labels of the class assignment matrix are correct as in the lower part of Table 2, the largest values will be on the diagonal of the class assignment matrix and smaller values are on the off-diagonals. Stated formally, there must be only one column maximum in each row (i.e., no ties for maximum in each row). In Table 3, the column maximum in row 1 is in column 2. The switched label detection algorithm simply moves row 1 to row 2. The column maximum in row 2 is in column 1, and the algorithm moves row 2 to row 1. The resulting locations of the

TABLE 3
Table 2 Repeated with Switched Labels

True	1	2	1	1	2	1	1	2	2	1
Assigned	2	1	1	2	2	1	2	1	1	2

	True 1	*True 2*
Assign 1	2	3
Assign 2	4	1

Note. The assigned row of the upper part has switched labels. Compare to the assigned row in the upper part of Table 2.

rows (2,1) are the corrected class labels. This approach is called the *column maxima switched label detection algorithm*, which is given as

1. { loop
2. l = row number of the kth column maximum
3. place l in the kth position of the new label vector
4. copy the lth row of M_{assign} to the kth row of $M_{\text{corrected}}$
5. } $k = 1 \ldots K$
6. return $M_{\text{corrected}}$ and the new label vector

where M_{assign} is the class assignment matrix such as the one given in the lower part of Table 3, K = the dimension of M_{assign}, and $M_{\text{corrected}}$ is an empty $K \times K$ matrix.

In general, this switched label detection algorithm will work when class assignment accuracy is moderately greater than chance, as detailed in the next section. In the sections that follow we review several other criteria the class assignment matrix must meet for the switched label detection algorithm to work.

Downloaded By: [University of California, Los Angeles] At: 20:58 9 February 2011

Class Assignment Accuracy

The first and most obvious criteria that must be met by the class assignment matrix is that class assignment must be sufficiently accurate such that each column contains only one row maximum. An example of a class assignment matrix with very high class assignment is given in the first row of Table 4.[1] The class assignment matrix is presented on the left, the corrected matrix is presented in the center with conditional proportion correct assignment for each class in the lower margin, and the new class labels are given on the right. In this ideal example, participants who have been given the label Class 2 (i.e., those in the second row of the assignment matrix) should have a corrected label of Class 3. Similarly, participants labeled as Class 3 should be relabeled as Class 2.

The first three rows in Table 4 show decreasing levels of conditional proportion correct class assignment. Conditional proportion correct assignment is the proportion of participants correctly assigned within each class, and we use this measure to formalize class assignment that is reasonably high, as alluded to in the previous section. A conditional proportion correct assignment is computed by dividing the number of participants correctly assigned to their true class n_{kCA} by the total number of participants in that class n_k, or $\frac{n_{kCA}}{n_k}$. In a class assignment matrix with correct label, this ratio will be the diagonal element of a given column divided by the total number of participants in that column. For example, 96/100 = .96 for the first class in the ideal example in Table 4.

In the ideal assignment example of Table 4 the proportion correct assignment within each class is .96, .91, and .89 for Classes 1, 2, and 3. Perfect class assignment, although unrealistic, would result in proportions correct assignment of 1 in each class. At the other extreme, random class assignment would result in proportions correct assignment of $\frac{1}{K}$ in each class, or .333 in this three-class example. At a minimum, conditional proportions correct assignment

[1]Note that the class sizes in Table 4 are equal. This is generally not the case in practice, but is used here for illustrative purposes.

TABLE 4
Example Switched and Unswitched Class Assignment Matrices

	Assignment Matrix			*Corrected Matrix*			*New Labels*
1. Ideal assignment	96	2	6	96	6	2	
	3	7	89	1	91	5	(1, 3, 2)
	1	91	5	3	7	89	
				.96	.91	.89	
2. Poor assignment	43	26	33	43	26	33	
	36	32	34	21	42	33	(1, 3, 2)
	21	42	33	36	32	34	
				.43	.42	.34	
3. Spurious correction	34	33	36	36	31	35	
	36	31	35	30	36	34	(3, 1, 2)
	30	36	34	34	33	36	
				.36	.36	.36	
4. Collapsed classes	3	3	4		Cannot be		
	92	96	3		corrected		(N/A, N/A, N/A)
	5	1	93				
				N/A	N/A	N/A	
5. Row detection	80	1	8	80	1	8	
	9	60	80	11	39	12	(1, 3, 2)
	11	39	12	9	60	80	
				.80	.39	.80	

Note. Example switched class assignment matrices. If switched labels can detected, the corresponding corrected class assignment matrix is given. Conditional proportion correct class assignment is given in the lower margin of corrected matrices. The new class labels are read as follows for the ideal assignment example: Class 1 parameters need the corrected label Class 1, Class 2 parameters need the corrected label Class 3, and Class 3 parameters need the corrected label Class 2.

must exceed $\frac{1}{K}$ in $K - 1$ classes[2] for the switched label detection algorithm to work. The minimum conditional proportion of correctly assigned participants allowed by the algorithm is incorporated in the class assignment criterion:

$$\mathrm{CA}_{crit} = \frac{\mathrm{CA}_{num}}{K}, \tag{4}$$

where CA_{num} is some number less than K and greater than 1. The value $1 - \mathrm{CA}_{num}$ is the proportion of participants greater than chance assignment desired by the user. For example, requiring 10% more participants to be correctly assigned than expected by chance requires a value of $\mathrm{CA}_{num} = 1.1$ while requiring 100% more participants to be correctly assigned than

[2]Note here that correctly labeling $K - 1$ classes will, by default, correctly label the kth class. Thus, the switched label detection algorithm requires that proportions correct assignment be greater than CA_{crit} in $K - 1$ rather than in K classes.

Downloaded By: [University of California, Los Angeles] At: 20:58 9 February 2011

expected by chance requires a value of $CA_{num} = 2$. For the three-class examples in Table 4, the corresponding values of the class assignment criterion would be $CA_{crit} = \frac{1.1}{3} = .367$ and $CA_{crit} = \frac{2}{3} = .667$.

To illustrate how selecting CA_{num} can address the problem of spuriously detecting switched labels, refer to Example 3 in Table 4. Here the proportions correct assignment are all .36. Using a value of $\frac{1.2}{K} = .4$ prevents the spurious detection of switched labels. Using $CA_{crit} = \frac{1.1}{3} = 0.367$ or assignment 10% greater than chance also prevents spurious detection of switched labels. However, if $CA_{crit} = \frac{1.05}{3} = 0.35$, requiring assignment to be only 5% greater than chance, switched labels would be detected by the algorithm. However, it is clear that assignment accuracy only 5% greater than chance does not engender a great deal of confidence in correcting class labels using class assignment information. More detail on selecting a value for CA_{num} is presented following the discussion on collapsing.

Collapsing

Example 4 in Table 4 shows a special case of poor assignment called collapsing. Collapsing is said to have happened when most of the participants in a class have been incorrectly assigned to one or more other classes. In other words, when collapsing has occurred there will be far fewer participants than expected in one class. In Example 4, most participants truly in Class 1 have been collapsed into Class 2 *if no switching has occurred*. If switching has occurred, Class 2 has been collapsed into Class 1, as shown here:

$$M_{collapsed} = \begin{bmatrix} 92 & 96 & 6 \\ 3 & 3 & 4 \\ 5 & 1 & 93 \end{bmatrix}.$$

There is no clear way to determine from the matrix alone whether label switching has occurred. In general, the rows of the class assignment matrix cannot be confidently corrected if it is (nearly) collapsed. However, the switched label detection algorithm presented thus far will still allow the matrix in Example 4 to be relabeled because assignment is still very good in $K - 1$ classes.

One way to address collapsing would be to require that all K classes have good class assignment, although this would reduce the number of situations in which the switched label detection algorithm would work. Instead, the switched label detection algorithm incorporates a collapsing criterion. The collapsing criterion is expressed as the proportion of participants the user will allow to be assigned into a single incorrect class. To illustrate, the assignment matrix in Example 4 is transformed by dividing each cell by its column total:

$$M_{collapsed}/n_k = \begin{bmatrix} .03 & .03 & .04 \\ .92 & .96 & .06 \\ .05 & .01 & .93 \end{bmatrix}.$$

Collapsing is detected when two entries in one row exceed the collapsing criterion. In Example 4, any collapsing criterion less than .92 will determine this matrix is collapsed. For example, if the criterion is set at .91, the user is allowing 91% of the participants in any class

Downloaded By: [University of California, Los Angeles] At: 20:58 9 February 2011

to be incorrectly assigned into a single class. On the other hand, a more conservative value of .5 might be selected, allowing only 50% of the participants in any one class to be incorrectly assigned into a single class. The collapsing criterion is expressed as

$$\text{CLPS}_{crit} = 1 - \frac{\text{CLPS}_{num}}{K}, \tag{5}$$

where CLPS_{num} is some value less than K and greater than 1, serving a function analogous to CA_{num} in the correct assignment criterion. The default value CLPS_{num} in the switched label detection algorithm is 1.2, meaning that the number of participants who can be incorrectly assigned to a single class cannot exceed 20% greater than chance. For $K = 3$, $1 - \frac{1.2}{3} = .6$, meaning that no more than 60% of participants can be incorrectly assigned to a single class. In practice, this default may be too conservative and smaller values of CLPS_{num} may be selected. Using $\text{CLPS}_{num} = 1.01$ allows the greatest number of participants to be incorrectly assigned into a single incorrect class. Using $\text{CLPS}_{num} = K$ requires that no participants in any one class be assigned into a single incorrect class (i.e., perfect class assignment). The `LBLinfo.r` script presented later allows the user to select a value CLPS_{num}. Selecting a value for CLPS_{num} is described in the next section.

Downloaded By: [University of California, Los Angeles] At: 20:58 9 February 2011

Selecting CA_{crit} and CLPS_{num}

The purpose of the CA_{crit} and CLPS_{num} criteria is to detect class assignment matrices that do not contain sufficient information for the switched label detection algorithm to provide trustworthy results. When class assignment is poor, or when assignment leads to collapsing, spurious relabeling can occur if the switched label detection algorithm is applied without checking for minimum levels of assignment accuracy and for collapsing.

In the simulation experience of the authors, data conditions often have estimation problems when conditional correct assignment is only 10% greater than chance. Estimation problems include model nonconvergence, infeasible parameter estimates that lead to nonpositive definite latent variable or residual variance matrices, nonpositive definite information matrices, or collapsing. Models with such problems should always be excluded from simulation summaries. If a condition has a substantial number of data sets with estimation problems, it must be concluded that such a data condition is not suitable for a given LVMM.

In the `R` implementation of the switched label detection algorithm presented here, the default values are $\text{CA}_{crit} = 1.2$ and $\text{CLPS}_{num} = 1.2$. A value of $\text{CA}_{crit} = 1.2$ indicates that correct assignment must be at least 20% greater than chance. For 2, 3, 4, and 5 classes, 20% greater than chance is .6, .4, .3, and .24. A value of $\text{CLPS}_{num} = 1.2$ indicates that the number of participants incorrectly assigned to a single class can be no more than 20% greater than chance.

The 20% above chance assignment criterion for CA_{crit} and CLPS_{num} was tested using simulated class assignment matrices. True and assigned class membership data were simulated from population *proportion* class assignment matrices that correspond to the following simulation conditions (the population proportion class assignment matrices are given in Appendix A): Four classes were used in each condition. Good, moderate, and poor assignment accuracies were examined. These three conditions were crossed with balanced class sizes and no collapsing, imbalanced class sizes and no collapsing, balanced class sizes and collapsing, and imbalanced

TABLE 5
Proportions of Data Sets in Which Switched Labels Could Not Be Corrected

	Bal/Not CLPS	*Imbal/Not CLPS*	*Bal/CLPS*	*Imbal/CLPS*
Big sample	$N = 400$	$N = 260$	$N = 400$	$N = 260$
Good	0.00	0.00	0.50	0.51
Moderate	0.01	0.04	0.50	0.51
Poor	0.75	0.78	0.78	0.80
Small sample	$N = 200$	$N = 150$	$N = 200$	$N = 150$
Good	0.00	0.00	0.50	0.52
Moderate	0.09	0.16	0.53	0.55
Poor	0.78	0.79	0.81	0.80

Note. Simulation results testing CA_{num} and $CLPS_{num}$ input values of 1.2, 1.15, and 1.1 for the switched label detection algorithm. Results in the table are for $CA_{num} = CLPS_{num} = 1.2$. Proportions are averaged over all possible class label permutations. Results for $CA_{num} = CLPS_{num} = 1.15$ and $CA_{num} = CLPS_{num} = 1.10$ are essentially the same, indicating stability in detecting switched labels that could be corrected.

Downloaded By: [University of California, Los Angeles] At: 20:58 9 February 2011

class sizes and collapsing conditions. Moderate and large sample sizes were used. In the balanced class size conditions, class sizes were (50, 50, 50, 50) and (100, 100, 100, 100). For the imbalanced class size conditions, class sizes were (50, 20, 50, 30) and (100, 20, 100, 40). Within each condition, data were simulated for all $4! = 24$ possible class label permutations. In total, there were 2 (sample sizes) $\times$ 3 (accuracies) $\times$ 2 (balanced or imbalanced class sizes) $\times$ 2 (not collapsed or collapsed) $\times = 24$ conditions, with each condition being crossed with all 24 possible class label permutations. A total of 500 data sets were simulated for each data condition and label permutation combination.

In every condition, the 20% above chance assignment criterion protected against the spurious detection of switched labels. To summarize the results, the portion of data sets for which labels could not be corrected were computed. Within each condition, these proportions were averaged across permutations, and are presented for $CA_{num} = CLPS_{num} = 1.2$ in Table 5. Larger proportions indicate conditions where observed class assignment matrices frequently contain insufficient information to inform class label correction.

To check the sensitivity of the criterion, $CA_{num} = CLPS_{num} = 1.15$ and $CA_{num} = CLPS_{num} = 1.10$ were also examined. The results were essentially the same as for $CA_{num} = CLPS_{num} = 1.2$. However, had the proportions in Table 5 dramatically increased from $CA_{num} = CLPS_{num} = 1.15$ to $CA_{num} = CLPS_{num} = 1.10$, this would indicate the lower criteria were too liberal. Similar sensitivity analyses are recommended in practice.

Column and Row Maxima

In the fifth example of Table 4, which is labeled `Row Detection`, the column maximum is in the second row for both Class 2 (60) and Class 3 (80). In this case, switched label detection might still be possible if there is only one row maximum in each column. This is the case for Example 5 in Table 4 where the row maxima for Classes 1, 2, and 3 are in columns 1 (80), 3 (80), and 2 (39). A modified switched label detection algorithm, called the *row maxima switched label detection algorithm*, is given here:

1. { loop
2. l = column number of the kth row maximum
3. place l in the kth position of the new label vector
4. copy the lth row of M_{assign} to the kth row of $M_{\text{corrected}}$
5. } $k = 1 \ldots K$
6. return $M_{\text{corrected}}$ and the new label vector.

Notice that in this example, the class assignment matrix can still be unswitched even though one column has poor class assignment (.39) relative to the other two. As noted earlier, the switched label detection algorithm will work if only $K - 1$ classes have reasonably high class assignment, and one class has poor assignment.

Downloaded By: [University of California, Los Angeles] At: 20:58 9 February 2011

STRENGTHS AND LIMITATIONS OF THE SWITCHED LABEL DETECTION ALGORITHM

Several limitations of the switched label detection algorithm are mentioned in the preceding sections. The algorithm is designed for data sets to which models with the correct number of classes are fitted to the data. In addition, reliable use of the algorithm requires that class assignment accuracy be reasonably high. These limitations might preclude the use of the switched label detection algorithm in some situations in which investigators might be interested. However, the switched label detection algorithm does not carry with it the limitations of arbitrary identifiability constraints. It also provides an advantage over inspection of parameter estimates. Although such a practice is highly encouraged and could easily be automated, the switched label detection algorithm can deal with situations in which parameter inspection might fail. For example, one set of parameters might suggest one set of new labels whereas a second set of parameters might suggest a different set of new labels. In this case, the switched label detection algorithm presented herein might still be able to reliably detect switched labels. Finally, the conditional proportions correct assignment required by the switched label detection algorithm are not overly restrictive. The default criterion of $\frac{1.2}{K}$ is associated with class assignment accuracy that would be considered relatively poor. Conditions with lower levels of correct assignment are likely to result in other problems such as nonconvergence or estimation errors like nonpositive definite latent variable matrices.

In the previous sections, the class assignment matrix is defined as a $K \times K$ square matrix. Although not necessarily immediately obvious, there are at least two situations where the assignment matrix might not be square. The first is when no participants are assigned to a given class, and the resulting assignment matrix will have a row of zeros. This can be seen as an extreme case of collapsing. The switched label detection algorithm implemented in the `LBLinfo.r` script will automatically exclude such matrices. The second situation is when a model is fit with greater or fewer classes than the number of classes in the simulated data. In this situation, the class assignment matrix is called a transition matrix. The transition matrix can be used to examine how $K - C$ class models combine participants into fewer classes or how $K + C$ class models redistribute participants into C extra classes. When models with the incorrect number of classes are fit, correct class assignment is ill defined, and the approach taken herein to detecting switched labels will not apply.

Downloaded By: [University of California, Los Angeles] At: 20:58 9 February 2011

AN R IMPLEMENTATION OF THE SWITCHED LABEL DETECTION ALGORITHM

The previous sections describe the theoretical development and implementation decisions associated with using the switched label detection algorithm. In this section we describe how to use the switched label detection algorithm as implemented in the `LBLinfo.r` script that can be downloaded from https://sourceforge.net/projects/lblinfo/. First, the free software R (R Development Core Team, 2008) is described. Second, the use of the `LBLinfo.r` script is illustrated with a single data set. Third, we show how the `LBLinfo.r` script can be used to automate obtaining the new label vector for multiple data sets, such as would be done in a simulation study. Fourth, we illustrate how the new label information can be used to correct parameter labels prior to summarizing across data sets, as was illustrated in Table 1. In the next section, an `R` script to automate label correction called `LBLcorrect.r` is introduced.

The R Language and Environment

R is a free, open source program for statistics and graphics (http://www.r-project.org/). Basic internet searches will find several Web sites providing R tutorials. First, R must be installed and the `LBLinfo.r` script must be downloaded and saved to a directory. After opening R the `LBLinfo.r` script must be sourced using `source('c:/my directory/LBLinfo.r')` where `c` in `c:/` should be changed to the appropriate drive letter, `my directory` should be changed to the full path of the directory in which the `LBLinfo.r` script is saved, and `LBLinfo.r` is the file name of the `LBLinfo.r` script. To do this, paste `source('c:/my directory/LBLinfo.r')` after the > in the R console. Note that R uses the forward slash (/) rather than the backslash (\) typically used in PC software. Sourcing an R script makes the functions in the script and the assignment matrices from Table 4 (named `eg1`, . . . , `eg6`) available in the current R session. To display the assignment matrix from the first example, type `eg1` after the > in the R console. After sourcing the `LBLinfo.r` script, set the working directory to the folder containing M*plus* simulation output using the `File > Change dir . . .` menu option.

Using the LBLinfo.r script with One Data Set

In this section, we illustrate the use of and output obtained from the `LBLinfo.r` script using the first example in Table 4. Assuming `R` is open and the script has been sourced as described in the previous section, run the `LBLinfo()` function by typing

```
LBLinfo(Mraw=eg1,CAnum=1.2,CLPSnum=1.2,K=3)
```

in the R console. The function's name is `LBLinfo()` and everything in the parentheses are inputs to the function. First, `Mraw` is the $K \times K$ class assignment matrix such as those in the left column of Table 4. The matrix eg1 is loaded into the working environment of `R` when `LBLinfo.r` is sourced. Instructions for inputting raw columns of true and assigned class membership are given in the next section. Second, `CAnum` is CA_{num} from Equation 4. Third, `CLPSnum` is CLPS_{num} from Equation 5. Finally, K is the number of classes.

The resulting output is printed to the R console. First, `CAnum`, `CLPSnum`, and K are printed along with `CAcrit` (computed using Equation 4) and `CLPScrit` (computed using Equation 5). Second, information on whether the input assignment matrix was square, whether any row or columns summed to zero, and whether the collapsing criterion `CLPScrit` was met. Then the input assignment matrix and the assignment matrix with corrected rows are printed if label correction is possible. Otherwise a matrix of `NA`s is produced. Next, a true/false indicator of whether the labels were correctable (`corrigible`) is printed. If `corrigible=TRUE`, all the criteria previously discussed have been met, and the labels of the parameter estimates from the data set can be corrected. Methods for correcting the labels of the parameter estimates are described later. If `corrigible=FALSE`, then the labels could not be corrected. Parameter estimates from the data set should be excluded from summaries across multiple data sets. Next, the new class labels and the conditional proportions correct assignment are printed. Finally, there is a true/false indicator of whether the labels were corrected, an indicator of whether the class assignment criteria was met in $K - 1$ classes, and a statement of whether the column or row maximum algorithm was used. Next we show how this script is used for multiple data sets.

Downloaded By: [University of California, Los Angeles] At: 20:58 9 February 2011

Using the LBLinfo.r with Multiple Data Sets

In this section, we illustrate how to use the `LBLinfo.r` script with five simulated data sets using output from M*plus*. The example data sets and associated M*plus* input and output can be downloaded from `https://sourceforge.net/projects/lblinfo/`. The data sets were generated according to a single factor, four-class finite mixture model. The factor variances were .5, 1, .75, and 1.25. The factor means were 4, −4, 8, and 0. The factor loadings were 1 for the first item and .8 for the remaining items, and residual variances were all .3. Measurement parameters were class invariant. The sample size was 942, with class proportions of .31, .331, .175, and .187 leading to expected sample sizes of 289, 312, 165, and 176. The data generating FMM was fit to these five data sets. To identify the model, the first factor loading in each class was fixed to 1 and the factor mean in the last class (Class 4) was fixed to 0. Intercepts and factor loadings were class invariant, and residual variances, factor variances, and factor means were specified to be class specific.

*Preparatory steps and description of M*plus *outputs.* First, a simulation study must be completed from which true and assigned class membership is saved. The generated data file must include true class membership, and this variable must be saved to an output file. This can be done using `IDVAR IS tc;` or `AUXILIARY IS tc;` in the `VARIABLE:` command of an M*plus* input file where `tc` is the name of the true class membership variable. The true and assigned class membership variables are saved using the `SAVEDATA:` command in M*plus*:

```
SAVEDATA:       Results are fmm_dat1.par;
FILE IS fmm_dat1.pro;
SAVE IS CPROBABILITIES;
```

where the `*.par` file contains the parameter estimates (see `RESULTS SAVING INFORMATION` at the end of the M*plus* output file for the order in which parameters and fit indexes are

saved). The *.pro file contains variables used in the analysis, ID and/or auxiliary variables, and posterior probabilities and assigned class membership. See SAVEDATA INFORMATION at the end of the M*plus* output file for the order of the variables; this can be used to determine assignCol and trueCol inputs for the LBLinfo.r script, as described later. The line SAVE IS CPROBABILITIES; tells M*plus* to save the class probabilities and assigned class membership variables to the *.pro file. Note that *.par and *.pro are arbitrary file extensions, and any file extension can be used.

After the simulation study has been completed, there should be one *.pro and *.par file for each data set/input file combination.[3] Though M*plus* (L. K. Muthén & Muthén, 2007) can run simulation studies internally using the Monte Carlo command, it *cannot* save the *.par and *.pro files. To avoid creating an input file for each data set and running each input manually, the M*plus* RUNALL utility can be used. The utility and instructions for its use are available at http://www.statmodel.com/runutil.shtml.

Next, the user will need to read the names of all of the *.pro files into R. One easy way to do this is to type the following in R:

```
prodir <- "c:\my directory\"
myfilenames <- list.files(prodir,glob2rx("*.pro"),full=FALSE)
```

where c:\my directory gives the full path of the directory containing the *.pro files. If an extension other than *.pro is used, this change must be reflected in the glob2rx function.

The LBLinfo.r function with multiple data sets. Next, the LBLinfo.r function can be run for several data sets using:

```
LBLinfo(filenames=myfilenames,assignCol=10,trueCol=5,CLPSnum=1.2,CAnum=1.2,
      K=4,fileprefix="mysim")
```

where myfilenames is as described previously. The assigned column number assignCol and true column number trueCol can be obtained by examining the M*plus* output file as described earlier. The files used in this example happen to have assigned and true class variables in columns 10 and 5, respectively. These files can be downloaded from https://sourceforge.net/projects/lblinfo/. The next three inputs CLPSnum, CAnum, and K were described in the previous section. When LBLinfo() is used with filenames instead of Mraw, results are saved to output files instead of being printed to the console. The fileprefix input gives a unique name for these output files. For example, if doing a sensitivity analysis with multiple values for CLPSnum and CAnum, users might specify fileprefix=mysim_CA1.2, fileprefix=mysim_CA1.15, and fileprefix=mysim_CA1.1, similar to the results summarized in Table 5.

[3]Note that if a model does not converge or has other fatal problems with estimation, *.pro and *.par files will not be produced by M*plus*. Also note that M*plus* will still produce *.pro and *.par files when latent variable matrices are not positive definite, when residual variance matrices are not positive definite, and when the first-order derivative matrix is not positive definite. These results are not trustworthy and should be excluded when taking summaries in a simulation study. This information is printed in the primary M*plus* output file *.out.

Downloaded By: [University of California, Los Angeles] At: 20:58 9 February 2011

Selected contents of mysim_RelabelSummary.txt

```
 class 1          class 2          class 3          class 4
 Min.   :0.99   Min.   :0.96   Min.   :1      Min.   :0.87
 1st Qu.:0.99   1st Qu.:0.97   1st Qu.:1      1st Qu.:0.92
 Median :0.99   Median :0.98   Median :1      Median :0.94
 Mean   :0.99   Mean   :0.98   Mean   :1      Mean   :0.93
 3rd Qu.:0.99   3rd Qu.:0.98   3rd Qu.:1      3rd Qu.:0.95
 Max.   :0.99   Max.   :0.98   Max.   :1      Max.   :0.97
```

Contents of mysim_ConditionalAssignment.txt

```
1 fmm_dat1.pro 0.986 0.984 1 0.942
1 fmm_dat2.pro 0.993 0.985 1 0.919
1 fmm_dat3.pro 0.99 0.977 1 0.946
1 fmm_dat4.pro 0.989 0.971 1 0.866
1 fmm_dat5.pro 0.993 0.963 1 0.97
```

Contents of mysim_RelabelValidation.txt

```
isSquare areRsum>0 areCsum>0 isNotClpsd K-1>tol? whichAlgo
TRUE TRUE TRUE TRUE TRUE colMax
TRUE TRUE TRUE TRUE TRUE colMax
TRUE TRUE TRUE TRUE TRUE colMax
TRUE TRUE TRUE TRUE TRUE colMax
TRUE TRUE TRUE TRUE TRUE colMax
```

Contents of mysim_NewClassLabels.txt

```
1 fmm_dat1.pro 2 1 3 4
1 fmm_dat2.pro 2 1 3 4
1 fmm_dat3.pro 1 3 2 4
1 fmm_dat4.pro 2 1 3 4
1 fmm_dat5.pro 2 1 3 4
```

FIGURE 1 Sample contents of output files produced by the LBLinfo.r script for the five simulated example data sets.

Downloaded By: [University of California, Los Angeles] At: 20:58 9 February 2011

LBLinfo() output files. Four output files are produced by the `LBLinfo()` function. First, the `mysim_RelabelSummary.txt` file summarizes the results of the switched label detection function across all data sets in `filenames`. This includes a printout of the inputs to the `LBLinfo()` function, counts of the number of data sets with correct labels, those with incorrect labels, those with incorrect labels that can be corrected (corrigible), and those with labels that could not be corrected (incorrigible). Then counts of data sets that met collapsing and correct assignment criteria are given, followed by a summary of conditional proportion correct assignments (as shown in the top part of Figure 1), and instructions on how to use the other output files. The second output file is `mysim_ConditionalAssignment.txt`, which has the conditional proportion correct assignments for each class for each data set (shown in the second part of Figure 1).

The third output file is `mysim_RelabelValidation.txt`, which shows for each data set whether the assignment matrix was square, whether row and column sums were all greater than zero, whether the collapsing criterion was met, whether the assignment criterion was met in

TABLE 6
Incorrect and Corrected Factor Variances

	Original Labels				*Corrected Labels*			
Data Set	*Class 1*	*Class 2*	*Class 3*	*Class 4*	*Class 1*	*Class 2*	*Class 3*	*Class 4*
`fmm_dat1`	0.86	0.24	0.43	1.31	0.24	0.86	0.43	1.31
`fmm_dat2`	0.86	0.20	0.45	1.59	0.20	0.86	0.45	1.59
`fmm_dat3`	0.24	0.49	0.90	1.20	0.24	0.90	0.49	1.20
`fmm_dat4`	0.98	0.24	0.66	4.07	0.24	0.98	0.66	4.07
`fmm_dat5`	1.07	0.26	0.49	1.30	0.26	1.07	0.49	1.30
M	0.80	0.29	0.59	1.90	0.24	0.93	0.50	1.90

Note. Incorrect and corrected factor variances for the five simulated data sets used to illustrated the `R` scripts `LBLinfo.r` and `LBLcorrect.r`.

Downloaded By: [University of California, Los Angeles] At: 20:58 9 February 2011

$K - 1$ classes, and which switched label detection algorithm was used, column or row (shown in the third part of Figure 1). The fourth output file is `mysim_NewClassLabels.txt`, which contains a binary indicator (1 = yes, 0 = no) whether labels could be corrected in the first column, the name of the file that contains the true and assigned class membership data, and the new class labels in the remaining columns (shown in the last part of Figure 1). These can be read into R as a data array for further use by typing `newLabels<-read.table('mysim_NewClassLabels.txt')` into R.

Correcting the Parameter Labels

In this section we illustrate how to use the new class labels shown in the bottom portion of Figure 1 to manually correct the labels of the parameter estimates. Issues associated with computer automation for correcting parameter labels are discussed in the next section. As described earlier, the population factor variances of the example data were .5, 1, .75, and 1.25. The estimated factor variances for the first data set were .86, .24, .43, and 1.31. The new class labels shown in the bottom part of Figure 1 are `2 1 3 4`. The new labels are read as follows: The parameter estimate for Class 1 (.861) should be given the new label Class 2. The parameter estimate for Class 2 (.244) should be given the new class label Class 1. The parameter estimate for Classes 3 and 4 are unchanged. After correcting the parameter estimate labels, the estimated factor variances are .24, .86, .43, and 1.31 for Classes 1 to 4, respectively. These results for the example data are summarized in Table 6.

ISSUES IN AUTOMATING PARAMETER LABEL CORRECTION

In practice, it is infeasible to manually correct the parameter labels using the new labels from the switched label correction algorithm. This section describes several issues that must be addressed when automating parameter label correction and presents an `R` script called `LBLcorrect.r`, which automates parameter label correction in some general LVMM conditions. Advice is given for automating parameter label correction under conditions more general than those automated by the `LBLcorrect.r` script.

Downloaded By: [University of California, Los Angeles] At: 20:58 9 February 2011

Parameter Constraints

The first thing that must be considered when correcting labels is whether a set of parameters is unconstrained or constrained. In the example data used in the previous section, the factor variances were unconstrained. That is, the factor variance is estimated in each class, and no constraints are placed on their estimation. When a set of parameters is unconstrained within class and are estimated in all K classes, automating parameter label correction is straightforward using computer if statements. For example, if the assigned label is 1 and the new label is 2, then move the estimate labeled 1 to the location for Class 2, and so on as described in the previous section. This is how the `LBLcorrect.r` script works for unconstrained parameters.

Simple if/then moving of parameter estimates from their assigned label to their corrected label also works for constrained parameter sets that meet two conditions. First, the constraint must be on the parameter in last class.[4] The two most common parameters of this type are the factor means and the class proportions.[5] Second, the last class must be correctly labeled. Notice in the bottom panel of Figure 1 that the last class, Class 4, is always correctly labeled. In the authors' experience, the last class is often correctly labeled for widely used LVMMs with three or more classes, although this is not guaranteed to hold in general. For models with two classes, incorrectly labeling the last class (Class 2) leads to the factor mean and logit class proportion in Class 1 being -1 times their correctly labeled estimate. This is easily fixed by again multiplying by -1.

For situations where the last class is not correctly labeled, or for other types of parameter constraints not discussed herein, parameter label correction and automation must be worked out on a case-by-case basis. To aid researchers who might find themselves in the situation where the factor mean of the last class is not correctly labeled, the label correction rules for the factor means in a three-class, one-factor model are given in Appendix B.

The LBLcorrect.r Script

The `LBLcorrect.r` function implements the simple if/then moving of parameter estimates from their assigned label to their corrected label described in the previous section. This is done using the information in the `*_NewClassLabels.txt` output file produced by the `LBLinfo.r` script. Using the example data described previously, we can implement the script using the following commands in R:

```
source("c:/my directory/LBLcorrect.r")
whichClass_v <- c(NA,NA,NA,
                  1,1,1,1,1,1,
                  2,2,2,2,2,2,
                  3,3,3,3,3,3,
                  4,4,4,4,4,
                  1,2,3)
```

[4]Fixing the parameter in a class other than the last class also works, but note that this is the default in M*plus* and a requirement of LBLcorrect script developed herein.

[5]One of the class proportions $(\pi_1, \ldots, \pi_K)$ is redundant because $\pi_K = 1 - \sum_{k=1}^{K-1} \pi_k$. In M*plus*, the logit class proportions $(\log(\pi_k/\pi_K))$ are estimated with the last class (K) as the reference class. These are called ALPHA(C) in TECHNICAL 1 in M*plus*.

```
whichType_v  <- c(NA,NA,NA,
                  1,2,3,4,5,6,
                  1,2,3,4,5,6,
                  1,2,3,4,5,6,
                  1,2,3,4,6,
                  7,7,7)
NfitInd <- 7
K=4
parnames <- list.files(getwd(),glob2rx("*.par"),full=F)
newLabelName <- "mysim_NewClassLabels.txt"
LBLcorrect(whichClass_v,whichType_v,NfitInd,parnames,
           newLabelName,K,fileprefix="mysim")
```

Downloaded By: [University of California, Los Angeles] At: 20:58 9 February 2011

The whichClass_v and whichType_v inputs combine to give a cross-classification of class label and parameter type for each parameter. The input whichClass_v designates the data generating label of each parameter in the order specified in TECHNICAL 1 M*plus* output. Any labels that are class invariant are specified as NA. In our example, the first three parameters are the factor loadings and are class invariant (i.e., they are measurement invariant). The six 1s in the second row of the example whichClass_v indicate that the next six parameters had the label Class 1 when the data were generated, and so forth for Classes 2, 3, and 4. The last row indicates that the last three parameters had the labels of Class 1, 2, and 3 when the data were generated.

The input whichType_v specifies a unique number for each type of parameter. As with whichClass_v, class-invariant parameters are specified as NA as is seen for the factor loadings in the first row for whichType_v. The second row corresponds to Class 1 and has the numbers 1 through 6. One, two, three, and four correspond to the class-specific error variances for Items 1 through 4, five corresponds to the factor mean in Class 1, and six corresponds to the factor variance in Class 1. The third row corresponds to Class 2 and also has the numbers 1 through 6, again corresponding to the four error variances, factor mean, and factor variance in Class 2. Notice that in the fifth row, the number five is missing, because the factor mean is constrained to zero in Class 4. The three 7s in the last row are for logit class proportions (see footnote 5).

The input NfitInd is the number of fit indexes, which can be counted by examining the "Order of data" subsection in the "RESULTS SAVING INFORMATION" information of an M*plus* output file. As in the LBLinfo() function, K is the number of classes, and parnames is the list of *.par files produced by M*plus* that contain parameter estimates, standard errors, and fit indexes. The input newLabelName is name of the *_NewClassLabels.txt file produced by the LBLinfo.r script. The output files produced by the LBLcorrect.r script are described in the script, and the outputs from the current example are included on the Web site.

Recall that when fitting LVMMs with continuous within-class factors, the factor mean(s) in the Kth class are fixed to zero as default in M*plus*. When $K = 2$, the factor mean(s) in Class 1 cannot be misestimated as the fixed value of zero in Class 2. When labels are switched, the factor mean(s) will take on a negative value. To fix this, the LBLcorrect.r script must be given the parameter number(s) of the factor mean(s) in the input value for whichIsFmean. For example, if the are three factors and two classes, and the parameter numbers in TECHNICAL 1 for the factor means are 10, 14, and 21, specify whichIsFmean=c(10,14,21). Additional details about the LBLcorrect.r script are given in the preamble to the script.

Computational Time

The `LBLinfo.r` and `LBLcorrect.r` scripts require the reading and writing of several files. As the number of files increases toward 1,000 or greater, the computational time can approach or exceed 60 seconds.

CONCLUSIONS

Mixture models are only identified up to permutation of class labels, resulting in the problem of switched class labels. In a simulation study, fitting a model to multiple data sets will result in arbitrary class labels, which complicates the computation of average parameter estimates and their standard errors. This article describes a switched class label detection algorithm that can provide the information needed to correct switched labels. Choices concerning a correct assignment criterion and a collapsing criterion for the algorithm were illustrated. The algorithm is implemented in the R script `LBLinfo.r`, which detects switched labels and reports corrected labels. This information can then be used to correct switched label parameters. The `LBLcorrect.r` script can use the output of the `LBLinfo.r` to automate parameter label correction and provide summaries for many common LVMMs. Conditions required by the switched label detection algorithm and the `LBLcorrect.r` script were noted and discussed.

Downloaded By: [University of California, Los Angeles] At: 20:58 9 February 2011

ACKNOWLEDGMENTS

We wish to thank the editor, the three anonymous reviewers, and Charles Laurin for helpful comments. The research of Stephen J. Tueller and Gitta H. Lubke was funded by R21 AG027360 from the National Institute on Aging (NIA). The research of Stephen J. Tueller was also supported by a Dissertation Research Award from The Society of Multivariate Experimental Psychology.

REFERENCES

Arminger, G., Stein, P., & Wittenberg, J. (1999). Mixtures of conditional mean- and covariance-structure models. *Psychometrika, 64,* 475–494.

Bollen, K. A. (1989). *Structural equations with latent variables.* New York: Wiley-Interscience.

Diebolt, J., & Robert, C. P. (1994). Estimation of finite mixture distributions through Bayesian sampling. *Journal of the Royal Statistical Society Series B: Methodological, 56,* 363–375.

Dolan, C. V., & van der Maas, H. L. J. (1998). Fitting multivariage normal finite mixtures subject to structural equation modeling. *Psychometrika, 63,* 227–253.

Everitt, B. S., & Hand, D. J. (1981). *Finite mixture distributions.* New York: Chapman & Hall.

Jedidi, K., Jagpal, H. S., & DeSarbo, W. S. (1997). STEMM: A general finite mixture structural equation model. *Journal of Classification, 14,* 23–50.

Jöreskog, K. G. (1966). Testing a simple structure hypothesis in factor analysis. *Psychometrika, 31,* 165–178.

Jöreskog, K. G. (1971). Simultaneous factor analysis in several populations. *Psychometrika, 36,* 409–426.

Lord, F. M. (1953). The relation of test score to the trait underlying the test. *Educational and Psychological Measurement, 13,* 517–549.

Lubke, G. H., & Neale, M. C. (2006). Distinguishing between latent classes and continuous factors: Resolution by maximum likelihood? *Multivariate Behavioral Research, 41,* 499–532.

Lubke, G. H., & Neale, M. C. (2008). Distinguishing between latent classes and continuous factors with categorical outcomes: Class invariance of parameters of factor mixture models. *Multivariate Behavioral Research, 43,* 592–620.

McLachlan, G., & Peel, D. (2000). *Finite mixture models.* New York: Wiley-Interscience.

Meredith, W., & Tisak, J. (1990). Latent curve analysis. *Psychometrika, 55,* 107–122.

Muthén, B. O. (2004). Latent variable analysis: Growth mixture modeling and related techniques for longitudinal data. In D. Kaplan (Ed.), *The Sage handbook of quantitative methodology for the social sciences* (p. 345–368). Thousand Oaks, CA: Sage.

Muthén, B. O., & Asparouhov, T. (2002). *Latent variable analysis with categorical outcomes: Multiple-group and growth modeling in M*plus (M*plus* Web Note No. 4). Retrieved December 2, 2010, from http://www.statmodel.com/examples/webnote.shtml

Muthén, B. O., & Asparouhov, T. (2006). Item response mixture modeling: Application to tobacco dependence criteria. *Addictive Behaviors, 31,* 1050–1066.

Muthén, B. O., Asparouhov, T., & Rebollo, I. (2006). Advances in behavioral genetics modeling using M*plus*: Applications of factor mixture modeling to twin data. *Twin Research and Human Genetics, 9,* 313–324.

Muthén, B. O., & Shedden, K. (1999). Finite mixture modeling with mixture outcomes using the EM algorithm. *Biometrics, 55,* 463–469.

Muthén, L. K., & Muthén, B. O. (2007). *M*plus *Version 5.0* [Computer program]. Los Angeles: Muthén & Muthén.

Neale, M. C., Aggen, S. H., Maes, H. H., Kubarych, T. S., & Schmitt, J. E. (2006). Methodological issues in the assessment of substance use phenotypes. *Addictive Behaviors, 31,* 1010–1034.

Nylund, K. L., Bellmore, A., Nishina, A., & Graham, S. (2007). Subtypes, severity, and structural stability of peer victimization: What does latent class analysis say? *Child Development, 78,* 1706–1722.

R Development Core Team. (2008). *R: A language and environment for statistical computing* [Computer software manual]. Vienna, Austria: Author.

Rost, J. (1988). Test theory with qualitative and quantitative latent variables. In R. Langeheine & J. Rost (Eds.), *Latent trait and latent class models* (pp. 147–171). New York: Plenum.

Sörbom, D. (1974). A general method for studying differences in factor means and factor structure between groups. *British Journal of Mathematical and Statistical Psychology, 27,* 229–239.

Titterington, D. M., Smith, A. F. M., & Makov, U. E. (1985). *Statistical analysis of finite mixture distributions*. New York: Wiley.

Torppa, M., Tolvanen, A., Poikkeus, A., Eklund, K., Lerkkanen, M., Leskinen, E., et al. (2007). Reading development subtypes and their early characteristics. *Annals of Dyslexia, 57*(1), 3–32.

Vermunt, J. K. (2003). Multilevel latent class models. *Sociological Methodology, 33*(1), 213–239.

Yung, Y. F. (1997). Finite mixtures in confirmatory factor-analysis models. *Psychometrika, 62,* 297–330.

Downloaded By: [University of California, Los Angeles] At: 20:58 9 February 2011

APPENDIX A
POPULATION PROPORTION ASSIGNMENT MATRICES USED TO TEST THE SWITCHED LABEL DETECTION ALGORITHM

Assignment Accuracy	*Noncollapsed*				*Collapsed*			
Good	.16	.04	.01	.01	.16	.04	.01	.01
	.05	.16	.04	.03	.05	.16	.04	.03
	.03	.04	.16	.05	.03	.04	.16	.16
	.01	.01	.04	.16	.01	.01	.04	.05
Moderate	.12	.06	.01	.01	.12	.06	.01	.01
	.07	.12	.06	.05	.07	.12	.06	.05
	.05	.06	.12	.07	.05	.06	.12	.12
	.01	.01	.06	.12	.01	.01	.06	.07
Poor	.08	.07	.04	.03	.08	.07	.04	.03
	.07	.08	.07	.06	.07	.08	.07	.06
	.06	.07	.08	.07	.06	.07	.08	.08
	.03	.04	.07	.08	.03	.04	.07	.07

APPENDIX B
THREE-CLASS MANUAL LABEL CORRECTION FOR FACTOR MEANS

Constrained parameters in the LVMM require careful attention when correcting labels. In this section, manual label correction is illustrated for the factor means in a three-class, one-factor model. The population values for the three factor means are 2, 1, and 0. When fitting the data generating model to the data, the factor mean in Class 3 is fixed to 0. In the following table, new class labels are presented for all possible switched label patterns. In the middle of the table, the hypothetical estimate values for Classes 1, 2, and 3 are given for each switched label pattern. On the right side of the table the label correction rule is given.

New Labels			*Estimated Values*			*Label Correction Rule*
1	2	3	2	1	0	Labels are correct
1	3	2	1	−1	0	Add 1: {1,−1,0} + 1 = {2,0,1}
2	1	3	1	2	0	Interchange the class labeled 2 with the class labeled 3[a]
2	3	1	−1	−2	0	Add 2: {−1,−2,0} + 2 = {1,0,2}
3	1	2	−1	1	0	Add 1: {−1,1,0} + 1 = {0,2,1}
3	2	1	−2	−1	0	Add 2: {−2,−1,0} + 2 = {0,1,2}

[a]The `LBLcorrect.r` script will correctly handle cases where the last class is correctly labeled.

Reading the table is illustrated using information from the second line of the table. The new class labels are 1, 3, and 2. The estimated Class 1 value has the correct label, but the estimate is incorrect. Estimated Class 2 should be relabeled as Class 3, and the estimated Class 3 should be relabeled as Class 2. The estimated factor means are 1, −1, and 0. To correct the factor means, add one to each estimated factor mean. This gives estimated factor means of 2, 0, and 1. Now factor means can be sorted according to the new labels, leading to the correct factor means of 2, 1, and 0.

Downloaded By: [University of California, Los Angeles] At: 20:58 9 February 2011