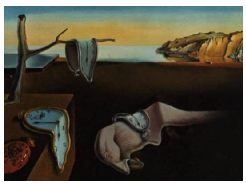

Journal of Statistical Software

October 2026, Volume 117, Issue 8. doi: 10.18637/jss.v117.i08

slca: An R Package for Structural Modeling for Multiple Latent Class Variables

Youngsun Kim
Seoul National University

Hwan Chung
Korea University

Abstract

This paper introduces the R package **slca**, a comprehensive tool designed to implement structural models for multiple latent class variables. The package facilitates the various methodologies for latent class analysis by offering advanced framework that enable to model complex relationships between multiple latent class variables. This paper details the theoretical framework underlying those latent class models, illustrates its implementation with **slca**, and demonstrates its versatility through real-world examples. By simplifying the construction for sophisticated latent class models, **slca** aims to make advanced statistical techniques accessible to researchers across various fields, thereby enhancing the analytical toolkit for exploring latent structures in data.

Keywords: latent class analysis, **Rcpp**, **slca**, structural latent class modeling.

1. Introduction

In the realm of behavioral and social sciences, understanding complex phenomena often requires delving beyond observable metrics to uncover the latent structures that govern individual behaviors and choices (Lazarsfeld and Henry 1968; McCutcheon 1987). Latent class analysis (LCA) is well-known categorical latent variable modeling that allows researchers to identify unseen groups within populations based on how individuals respond to certain variables. This methodology has propelled research in various disciplines, offering nuanced insights into the hidden heterogeneities within populations.

However, as research questions have become more sophisticated, the limitations of traditional LCA in addressing the multifaceted nature of human behavior have become apparent. A significant limitation is that traditional LCA derives only a single latent variable, which often fails to capture the complexity of many psychological, educational, and social processes (Lanza, Bray, and Collins 2012; Nylund, Asparouhov, and Muthén 2007). Moreover, the

static nature of traditional LCA fails to account for the dynamic evolution of behaviors and attitudes over time, a key aspect of human development and social change.

For instance, in behavioral research, understanding mental health disorders might require simultaneously considering substance use behavior and violent behavior, each of which could be conceptualized as separate but interrelated latent class variables. Additionally, tracking these aspects over time can reveal critical insights into the progression of disorders, the effectiveness of interventions, and the impact of life events, necessitating models that can analyze longitudinal data within the latent class framework.

This recognition has driven the development of statistical techniques that extend beyond the traditional LCA framework to accommodate multiple latent class variables. Latent transition analysis (LTA; Collins and Lanza 2009) also known as a latent Markov model (LMM), offers a dynamic perspective by examining changes in latent class memberships over time, providing insights into how individuals transition between different states. While latent class profile analysis (LCPA; Chung, Anthony, and Schafer 2010) also examines longitudinal pattern of the latent class memberships, it does so by demonstrating the dynamic shifts in latent class membership through a hierarchical structure, where the latent class memberships of each state are summarized by a higher-level latent class variable known as the latent class profile. Similarly, joint latent class analysis (JLCA; Jeon, Lee, Anthony, and Chung 2017) introduces a higher-level latent class variable that describes adolescent delinquency behavioral patterns by summarizing four latent class variables: violent behavior, alcohol consumption, cigarette smoking, and other drug use. In addition to these extended latent class models, there can be additional latent class variables or layers of latent structure. For instance, Lee and Chung (2017) investigated the effect of drug-use behavior on violent behavior, by appending additional latent class variable to JLCA. Drug-use behavior, a joint pattern of cigarette smoking, alcohol consumption, and other illegal drug use, is measured by JLCA and they examined how the latent class membership of violent behavior changes in response to drug-use behavior. Moreover, Kim, Chung, and Jeon (2020) explored more complex data involving repeatedly measured multidimensional data by combining JLCA with LTA and LCPA. The latent attribute of adolescent emotional well-being is assessed by the joint pattern of two different latent class variable, depression and anxiety subtypes, with the dynamic change of the joint pattern captured using LTA or LCPA methods.

Several R (R Core Team 2026) packages have been developed to facilitate the application of LCA. The **poLCA** package (Linzer and Lewis 2011) provides a user-friendly interface for standard LCA with polytomous outcome variables. Other packages, such as **BayesLCA** (White and Murphy 2014) and **LCAvarsel** (Fop and Murphy 2018), offer specialized functionalities like Bayesian estimation or variable selection in LCA. However, these packages are limited to handling only a single latent class variable, hindering their ability to capture the complexity of many real-world phenomena involving multiple interrelated latent class variables. The **LMest** package (Bartolucci, Pandolfi, and Pennoni 2017), **glca** package (Kim, Jeon, Chang, and Chung 2022) and **multilevLCA** package (Lyrvall, Mari, Bakk, Oser, and Kuha 2025) stand out from other R packages as they allow for the analysis of multiple latent class variables. However, these packages are specifically designed for particular latent structures. **LMest** is specialized for LTA in repeated measures data, while **glca** and **multilevLCA** are specialized for multilevel LCA in grouped data. These packages accommodates the analysis of multiple latent class variables that are not part of repeatedly measured data. This limitation restricts its applicability to a broader range of complex latent structures.

In the broader context of latent variable modeling, structural equation modeling (SEM) has been a popular approach for analyzing complex relationships among latent constructs. R packages such as **lavaan** (Rosseel 2012) and **OpenMx** (Boker *et al.* 2026) provide powerful tools for SEM. However, these packages primarily focus on continuous latent variables and do not easily accommodate categorical latent variables, which are the focus of LCA and its extensions.

Commercial software such as **Mplus** (Muthén and Muthén 1998) and **Latent GOLD** (Vermunt and Magidson 2005) provides a broader range of latent class models than existing R packages. However, their high costs, limited flexibility for customization, and closed-source nature restrict accessibility and adaptability for further research.

Recognizing the need for a comprehensive tool that can seamlessly integrate advanced LCA methodologies, we introduce the **slca** package (Kim and Chung 2026) for R. This package is designed to simplify the application of structural modeling with multiple latent class variables, offering researchers a versatile platform for conducting sophisticated LCAs. The **slca** package not only makes these advanced modeling techniques accessible to a broader audience but also provides rapid estimating procedure using **Rcpp** (Eddelbuettel and François 2011). The **slca** package (Kim and Chung 2026) is available from the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/package=slca`.

Through detailed documentation and examples, this paper describes how **slca** facilitates the exploration of latent structures and their dynamics. This paper streamlines the model specification process with an intuitive syntax, allows for efficient parameter estimation using the expectation-maximization (EM) algorithm and the upward-downward method, and provides a range of tools for model diagnosis and post-hoc research.

2. Model

2.1. Structural latent class model

The model handled by the **slca** package encompasses diverse latent class models, but there are some assumptions for the model structure. First, each variable depends only on one variable, meaning that each variable has a single parent variable. The parent variable of X_j is denoted as $\text{pa}(X_j)$. The terms *parent* and *child* variables used here correspond to what is usually referred to in the SEM framework as exogenous and endogenous variables, respectively. Second, the variables are conditionally independent given their parent variables. For example, if a latent class variable X is measured by M manifest variables, $Y_1, \ldots, Y_M$ (i.e., the parent variable of Y_m is the latent class variable X for $m = 1, \ldots, M$), then these manifest variables are mutually independent given the class membership of X. Similarly, if a latent class variable X_0 affects the other latent class variables $X_1, \ldots, X_p$ (i.e., the parent variable of $X_1, \ldots, X_p$ is the latent class variable X_0), then these latent class variables are mutually independent given the class membership of X_0.

If the variable X_r has no parent, it is referred to as a *root variable*, and the *root prevalence* is parameterized as π and helps us understand the baseline distribution of the latent classes:

$$\pi_k^{(r)} = \mathsf{P}(X_r = k)$$

Note that this is similar to the initial probability in a hidden Markov model. If the variable

X_j is latent and has a parent variable, then the conditional probability is parameterized as follows:

$$\tau_{k|l}^{(j)} = \mathsf{P}(X_j = k \mid \text{pa}(X_j) = l)$$

This conditional probability, referred to as a *transition probability*, describes the probability of an individual belonging to a particular latent class, given their membership in parent latent class variable. For instance, in LTA, it reveals the probability of being in each latent state at a later time point, given the latent state at an earlier time point. In JLCA, it illustrates the probability of belonging to the membership of each latent class variable (e.g., smoking, alcohol use, or other drug-use behavior), given the membership of joint latent class variable (e.g., overall substance use behavior).

Using single parent and conditional independence assumptions, we can construct the following likelihood function of latent variables $\mathbf{X} = \{X_1, \ldots, X_p\}$ by sequentially multiplying the conditional probabilities as follows:

$$\begin{aligned}\mathsf{P}(\mathbf{X} = \mathbf{x}) &= \prod_{r \in R} \mathsf{P}(X_r = x_r) \prod_{j \in L} \mathsf{P}(X_j = x_j \mid \text{pa}(X_j) = \text{pa}(x_j)) \\ &= \prod_{r \in R} \pi_{x_r}^{(r)} \prod_{j \in L} \tau_{x_j|\text{pa}(x_j)}^{(j)},\end{aligned}$$

where $\mathbf{x} = \{x_1, \ldots, x_p\}$ denote the possible states of the latent class variables $\mathbf{X}$. Here, R represents the index set for root latent class variables, while L denotes the index set for latent class variables with parent, respectively.

For the observed variable Y_m, the conditional probability is referred to as *conditional response probability*. This probability, parameterized as ρ, describes how observations respond according to latent class membership:

$$\rho_{r|k}^{(m)} = \mathsf{P}(Y_m = r \mid \text{pa}(Y_m) = k)$$

Thus, by examining the ρ-parameters, we can interpret the meaning of each latent class variable. Using these model parameters, we can reparameterize the complete-data likelihood function with M manifest items, $\mathbf{Y} = \{Y_1, \ldots, Y_M\}$:

$$\begin{aligned}\mathsf{P}(\mathbf{X} = \mathbf{x}, \mathbf{Y} = \mathbf{y}) &= \mathsf{P}(\mathbf{X} = \mathbf{x})\mathsf{P}(\mathbf{Y} = \mathbf{y} \mid \mathbf{x}) \\ &= \prod_{r \in R} \mathsf{P}(X_r = x_r) \prod_{j \in L} \mathsf{P}(X_j = x_j \mid \text{pa}(x_j)) \prod_{m=1}^{M} \mathsf{P}(Y_m = y_m \mid \text{pa}(y_m)) \\ &= \prod_{r \in R} \pi_{x_r}^{(r)} \prod_{j \in L} \tau_{x_j|\text{pa}(x_j)}^{(j)} \prod_{m=1}^{M} \rho_{y_m|\text{pa}(y_m)}^{(m)},\end{aligned} \tag{1}$$

where $\mathbf{y} = \{y_1, \ldots, y_M\}$ represent the outcomes of manifest variables $\mathbf{Y}$. In (1), $\text{pa}(x_j)$ and $\text{pa}(y_m)$ denote the realized values of $\text{pa}(X_j)$ and $\text{pa}(Y_m)$, respectively. Since the latent class variables are not observed, they must be summed out from this complete-data likelihood in order to obtain the observed-data likelihood:

$$\mathsf{P}(\mathbf{Y} = \mathbf{y}) = \sum_{\mathbf{x}} \mathsf{P}(\mathbf{X} = \mathbf{x}, \mathbf{Y} = \mathbf{y})$$

2.2. Existing models specified by structural latent class model

The general model equation given in (1) can be specialized to represent various well-known latent class models. This is achieved by defining the specific set of latent and manifest variables for a given model and specifying the parent-child relationships among them. The following examples specialize the equation onto existing models.

For a traditional LCA, the model consists of a single latent variable X and a set of M manifest items $\mathbf{Y} = \{Y_1, \ldots, Y_M\}$ (i.e., $\text{pa}(Y_m) = X$ for $m = 1, \ldots, M$). The structure of LCA denotes that a single latent variable X is the root variable (i.e., $\text{pa}(X) = \emptyset$). Applying this structure to the general likelihood formula given in (1) yields the familiar LCA model:

$$\mathsf{P}(X = x, \mathbf{Y} = \mathbf{y}) = \mathsf{P}(X = x) \prod_{m=1}^{M} \mathsf{P}(Y_m = y_m \mid x)$$

The LTA, which is a type of latent Markov model, involves a time-ordered sequence of latent variables $\mathbf{X} = \{X_1, \ldots, X_T\}$, where X_t is measured by a set of M manifest items $\mathbf{Y}_t = \{Y_{1t}, \ldots, Y_{Mt}\}$ at time t (i.e., $\text{pa}(Y_{mt}) = X_t$ for $m = 1, \ldots, M$) for $t = 1, \ldots, T$. The structure assumes a first-order Markov property, defining the parent sets as $\text{pa}(X_1) = \emptyset$ and $\text{pa}(X_t) = X_{t-1}$ for $t = 2, \ldots, T$. Then, applying this structure to the general likelihood formula given in (1) yields the complete-data likelihood of the LTA model:

$$\mathsf{P}(\mathbf{X} = \mathbf{x}, \mathbf{Y} = \mathbf{y}) = \mathsf{P}(X_1 = x_1) \prod_{t=2}^{T} \mathsf{P}(X_t = x_t \mid x_{t-1}) \prod_{m=1}^{M} \mathsf{P}(Y_{mt} = y_{mt} \mid x_t) \tag{2}$$

Hierarchical models like JLCA and LCPA feature a higher-order latent variable. A latent variable X_0, that is a parent to several first-order latent variables, $\mathbf{X} = \{X_1, \ldots, X_J\}$, where X_j is measured by a set of M_j manifest items $\mathbf{Y}_j = \{Y_{1j}, \ldots, Y_{M_j j}\}$ (i.e., $\text{pa}(Y_{mj}) = X_j$ for $m = 1, \ldots, M_j$) for $j = 1, \ldots, J$. The primary structural difference between JLCA and LCPA often lies in the interpretation and constraints; LCPA typically implies the X_j variables are repeated measures of the same construct over time, often with measurement invariance imposed. For both, the fundamental structure is defined by the parent sets $\text{pa}(X_0) = \emptyset$ and $\text{pa}(X_j) = X_0$ for $j = 1, \ldots, J$. The resulting complete-data likelihood is

$$\mathsf{P}(\mathbf{X} = \mathbf{x}, \mathbf{Y} = \mathbf{y}) = \mathsf{P}(X_0 = x_0) \prod_{j=1}^{J} \mathsf{P}(X_j = x_j \mid x_0) \prod_{m=1}^{M_j} \mathsf{P}(Y_{mj} = y_{mj} \mid x_j). \tag{3}$$

A third-order hierarchical LCA model, JLCPA has a complex and longitudinal structure. The model assumes a top-level, joint latent profile variable, X_0, which influences a set of second-order joint class variables, $\mathbf{X}_0 = \{X_{01}, \ldots, X_{0T}\}$, existing at each time point t (for $t = 1, \ldots, T$). Each second-order joint class variable at a given time point, X_{0t} serves as the parent to a set of J latent class variables, $\mathbf{X}_t = \{X_{1t}, \ldots, X_{Jt}\}$, at that same time point. Finally, each of these latent class variables, X_{jt}, is measured by a set of M_j observable manifest items, $\mathbf{Y}_{jt} = \{Y_{1jt}, \ldots, Y_{M_j jt}\}$. This structure is defined by the parent sets $\text{pa}(X_0) = \emptyset$, $\text{pa}(X_{0t}) = X_0$, $\text{pa}(X_{jt}) = X_{0t}$, and $\text{pa}(Y_{mjt}) = X_{jt}$ for $m = 1, \ldots, M_j$, $j = 1, \ldots, J$ and $t = 1, \ldots, T$. Applying this complex hierarchical and longitudinal structure to the general

likelihood formula yields the model's complete-data likelihood:

$$\mathsf{P}(\mathbf{X} = \mathbf{x}, \mathbf{Y} = \mathbf{y}) = \mathsf{P}(X_0 = x_0) \times \prod_{t=1}^{T} \mathsf{P}(X_{0t} = x_{0t} \mid x_0) \prod_{j=1}^{J} \mathsf{P}(X_{jt} = x_{jt} \mid x_{0t}) \prod_{m=1}^{M_j} \mathsf{P}(Y_{mjt} = y_{mjt} \mid x_{jt})$$

2.3. Incorporating exogenous covariates

The structural latent class model can be extended to include exogenous covariates (Z) to explain variations in the membership probabilities of each latent class variable (X). This allows researchers to investigate how individual characteristics predict latent class membership. The relationship is specified using a multinomial logistic regression model.

For estimating covariate effects, a three-step approach is adopted for this package instead of one-step approach. While the one-step approach is a valid alternative, it has several practical disadvantages. It can be cumbersome in exploratory studies with many potential covariates, as the entire measurement model must be re-estimated each time a variable is changed. Furthermore, the three-step framework better aligns with the logical workflow of many applied researchers, who prefer to first establish a stable classification model before examining its external predictors. The specific procedure for implementing this three-step estimation, including methods to correct for known biases, is detailed in Section 3.6.

3. Model estimation and inference

3.1. EM algorithm

To estimate the model parameters, we employ the expectation-maximization (EM) algorithm (Dempster, Laird, and Rubin 1977). The EM algorithm is an iterative method for finding maximum likelihood estimates when the model involves unobserved latent variables, making it ideal for structural latent class models. The algorithm iterates sequential two steps until convergence: an expectation-step (E-step) and a maximization-step (M-step).

In the E-step, we define Q function as the expected value of the log-likelihood function with respect to the conditional distribution of latent class variables given the observed data and the current parameter estimates:

$$Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(t)}) = E_{\mathbf{X}|\mathbf{Y},\boldsymbol{\theta}^{(t)}} \left[\log L(\boldsymbol{\theta}; \mathbf{X}, \mathbf{Y})\right]$$

Here, $\boldsymbol{\theta}$ denotes the collection of the parameters, $\boldsymbol{\pi}$, $\boldsymbol{\tau}$, and $\boldsymbol{\rho}$, and $\boldsymbol{\theta}^{(t)}$ denotes the current estimates of those parameters. To formulate the Q function, we need to calculate the conditional probability of the combination of latent class variables for each observation:

$$\mathsf{P}\left(\mathbf{X} = \mathbf{x} \mid \mathbf{y}_i; \boldsymbol{\theta}^{(t)}\right) = \frac{\mathsf{P}(\mathbf{X} = \mathbf{x}, \mathbf{Y} = \mathbf{y}_i; \boldsymbol{\theta}^{(t)})}{\mathsf{P}(\mathbf{Y} = \mathbf{y}_i; \boldsymbol{\theta}^{(t)})}$$

In the M-step, we maximize the Q function with respect to the model parameters. Using the Lagrange multiplier, the parameters are updated by averaging the posterior probabilities as

follows:

$$\begin{aligned}
\pi_k^{(r)} &= \frac{\sum_{i=1}^n \mathsf{P}(X_r = k \mid \mathbf{y}_i)}{n} \\
\tau_{k|l}^{(j)} &= \frac{\sum_{i=1}^n \mathsf{P}(X_j = k, \mathrm{pa}(X_j) = l \mid \mathbf{y}_i)}{\sum_{i=1}^n \mathsf{P}(\mathrm{pa}(X_j) = l \mid \mathbf{y}_i)} \\
\rho_{r|k}^{(m)} &= \frac{\sum_{i=1}^n \mathsf{P}(\mathrm{pa}(Y_m) = k \mid \mathbf{y}_i) \times I(y_{im} = r)}{\sum_{i=1}^n \mathsf{P}(\mathrm{pa}(Y_m) = k \mid \mathbf{y}_i)}
\end{aligned}$$

In the M-step, summing out irrelevant variables from the full joint probability incurs high computational costs. However, we notice that the full joint posterior probability of all latent class variables are not required; instead, we only need the marginal posterior probabilities of each latent class variable or the joint posterior probabilities of each latent class variable with its parent variable. Therefore, we compute these marginal or partial joint posterior probabilities directly using the upward-downward algorithm (Crouse, Nowak, and Baraniuk 1998), and update the model parameters accordingly. The detailed procedure of deriving the update formula for each parameter and the upward-downward algorithm is provided in Appendix A and Appendix B.

3.2. Hybrid algorithm

Although the upward-downward algorithm makes the EM algorithm feasible and more efficient, the EM algorithm can take more time to achieve complete convergence. The Newton-Raphson (NR) algorithm can be considered for faster estimation of model parameters, but it is highly sensitive to initial estimates (Schnabel, Koonatz, and Weiss 1985). If the initial values are not properly chosen, the NR algorithm may fail to converge and can even be unable to proceed due to the singularity of the Hessian matrix.

To address these issues, we can combine the strengths of both algorithms by initiating the estimation with the EM algorithm and finishing with the NR algorithm. The EM algorithm's stability and insensitivity to initial values provide reliable initial estimates, minimizing the risk of singularities. Once the parameters are reasonably estimated by the EM algorithm, switching to the NR algorithm can expedite convergence, leveraging on its faster convergence rate near the optimum. This hybrid approach mitigates the unstable initial value problem of the NR algorithm and the slow convergence problem of the EM algorithm.

In **slca**, we can choose the estimation algorithm by specifying `method` argument in the `estimate` function. The available options are `"em"`, `"nlm"`, and `"hybrid"`, corresponding to the EM, the NR using R's `nlm` function (Dennis and Schnabel 1996) and the hybrid algorithm described above, respectively. Additionally, within the hybrid option, you can configure how long the EM algorithm should be used before switching to the NR algorithm by adjusting the `control` options (see `slcaControl()`).

3.3. Standard error calculation

To assess the uncertainty of the estimator $\hat{\boldsymbol{\theta}}$, we report standard errors obtained from the empirical information matrix (Berndt, Hall, Hall, and Hausman 1974). This approach, also known as the outer-product-of-gradients (OPG), approximates the Fisher information matrix

as follows:

$$I(\hat{\boldsymbol{\theta}}) \approx \sum_{i=1}^{n} \mathbf{s}_i(\hat{\boldsymbol{\theta}}) \mathbf{s}_i(\hat{\boldsymbol{\theta}})^\top$$

where $\mathbf{s}_i(\hat{\boldsymbol{\theta}})$ is the score vector (the gradient of the individual log-likelihood function) evaluated at the final parameter estimates.

In **slca** models, the parameters are probabilities and thus subject to constraints (e.g., being in $(0, 1)$ and summing to one), which cause the full information matrix to be singular. To resolve this, we reparameterize the model. We transform the original constrained parameters $\boldsymbol{\theta}$ (i.e., class prevalences $\boldsymbol{\pi}$, conditional dependencies $\boldsymbol{\tau}$, and item responses $\boldsymbol{\rho}$) into a set of unconstrained, independent free parameters (denoted as $\boldsymbol{\psi}$, $\boldsymbol{\zeta}$, and $\boldsymbol{\omega}$, respectively) using a logit transformation. The standard errors are then computed on the scale of these free parameters, which we collectively denote as $\boldsymbol{\eta} = \{\boldsymbol{\psi}, \boldsymbol{\zeta}, \boldsymbol{\omega}\}$, and transformed back. The procedure is as follows: first, the information matrix for the free parameters, $I(\hat{\boldsymbol{\eta}})$, is computed using the scores $\mathbf{s}_i(\boldsymbol{\eta}) = \partial \log L_i / \partial \boldsymbol{\eta}$. Second, this matrix is inverted to obtain the variance-covariance matrix $\mathsf{VAR}(\hat{\boldsymbol{\eta}})$. Third, the variance on the original parameter scale is obtained using the Jacobian matrix $\mathbf{J} = \partial \boldsymbol{\theta} / \partial \boldsymbol{\eta}^\top$ as follows:

$$\mathsf{VAR}(\hat{\boldsymbol{\theta}}) \approx \mathbf{J}\, \mathsf{VAR}(\hat{\boldsymbol{\eta}})\, \mathbf{J}^\top.$$

The detailed formulas for the score vector $\mathbf{s}_i(\boldsymbol{\eta})$ and the Jacobian matrix $\mathbf{J}$ are provided in Appendix C.

3.4. Estimation issues

Missing data

Missing data is a common issue in empirical research. In our framework, we handle missing data based on two scenarios. First, if all manifest variables for an observation i (i.e., $\mathbf{y}_i$) are missing, that observation provides no information for model estimation. Therefore, it is excluded from the analysis via listwise deletion. For observations with only partial missing data, we adopt the missing at random (MAR) assumption. MAR posits that the probability of a value being missing for a particular manifest variable depends only on the observed information, not on the unobserved value itself. The EM algorithm provides a natural framework for handling MAR data, as it effectively integrates over the distribution of the missing values. The modifications to the E-step and M-step are as follows.

In the E-step, the posterior probability of a latent class configuration for an observation i is calculated using only its observed manifest variables. The likelihood contribution of observation i is computed over these observed variables, and the posterior probability is subsequently derived as

$$\mathsf{P}\left(\mathbf{X} = \mathbf{x} \mid \mathbf{y}_i^{\text{obs}}; \boldsymbol{\theta}^{(t)}\right) = \frac{\mathsf{P}(\mathbf{X} = \mathbf{x}; \boldsymbol{\theta}^{(t)}) \mathsf{P}(\mathbf{Y}_i = \mathbf{y}_i^{\text{obs}} \mid \mathbf{x}; \boldsymbol{\theta}^{(t)})}{\mathsf{P}(\mathbf{Y}_i = \mathbf{y}_i^{\text{obs}}; \boldsymbol{\theta}^{(t)})},$$

where the marginal probability in the denominator is $\mathsf{P}(\mathbf{Y}_i = \mathbf{y}_i^{\text{obs}}; \boldsymbol{\theta}^{(t)}) = \sum_{\mathbf{x}} \mathsf{P}(\mathbf{X} = \mathbf{x}; \boldsymbol{\theta}^{(t)}) \mathsf{P}(\mathbf{Y}_i = \mathbf{y}_i^{\text{obs}} \mid \mathbf{x}; \boldsymbol{\theta}^{(t)})$.

In the M-step, the update formulas for the latent structure parameters, $\boldsymbol{\pi}$ and $\boldsymbol{\tau}$, remain structurally the same, though they use the posterior probabilities computed from the observed

data as described above. However, the update for the conditional response probability, $\rho_{r|k}^{(m)}$, requires a specific adjustment. Let O_m be the set of observations for which item Y_m is observed, and M_m be the set for which Y_m is missing. For an observation $i \in O_m$, its contribution to the expected count for a specific response category y_m is weighted by its posterior probability, as indicated by the term $I(Y_{im} = r)$. For an observation $i \in M_m$, the response is unknown. Therefore, its expected contribution is distributed across all possible response categories of item m according to the current parameter estimates $\boldsymbol{\rho}^{(t)}$ from the previous iteration. The updated formula for $\rho_{r|k}^{(m)}$ is

$$\rho_{r|k}^{(m)^{(t+1)}} = \frac{\sum_{i \in O_m} \phi_{i,k}^{(m)} I(Y_{im} = r) + \sum_{i \in M_m} \phi_{i,k}^{(m)} \rho_{r|k}^{(m),(t)}}{\sum_{i=1}^{n} \phi_{i,k}^{(m)}},$$

where $\phi_{i,k}^{(m)} = \mathsf{P}(\text{pa}(Y_{im}) = k \mid \mathbf{y}_i)$. This approach ensures that all available information is used to update the parameters, providing robust estimates in the presence of missing data under the MAR assumption.

Identifiability

As with other statistical models, ensuring model identifiability is fundamental in latent class models. This concern becomes particularly relevant for the models handled in the **slca** package, as their high-dimensional parameter spaces increase the likelihood of encountering such problems. A necessary condition for a model to be theoretically identifiable is that the number of free parameters does not exceed the degrees of freedom provided by the observed data. A model that violates this rule is considered underidentified, implying that a unique set of parameters that best fits the data may not exist.

Empirically, identifiability issues are diagnosed via the Fisher information matrix, from which the variance-covariance matrix of the estimators is obtained. For a well-identified model, the information matrix is positive definite and thus invertible as discussed by McHugh (1956) and Goodman (1974). However, in cases of empirical underidentification–often due to sparse data or poor class separation–the log-likelihood surface may be flat in certain dimensions. This yields an information matrix that is not positive definite, and consequently singular. As the standard errors of the parameter estimates are derived from the inverse of this matrix, a singularity results in infinitely large or non-computable standard errors. The presence of such standard errors is therefore a clear indication that the parameter estimates are unstable and unreliable.

Local maximum problem

The EM algorithm, while stable and effective at finding an optimum, may converge to a local maximum rather than the global maximum. A local maximum is a point where the function value is higher than at nearby points, but there may exist other points further away with even higher values. This means the algorithm might find a solution that is optimal in a small region of the parameter space, but not the best possible solution overall. To mitigate this issue, it is common to run the EM algorithm with multiple initial values. Starting the algorithm from various points in the parameter space, the chances of finding the global maximum increase, as it reduces the likelihood of getting trapped in local maxima.

In the **slca** package, you can address the issue of local maxima by using the `nrep` option within the `slcaControl` argument to specify the number of repetitions, each with different initial values. The initial values for the π parameters are generated to be equal (e.g., for a two-class variable, the proportions are both 0.5), while for the τ and ρ parameters, a small perturbation is applied to uniform probabilities. These generated probabilities are then normalized to sum to one, conditional on each state of the parent variable. Additionally, the `test.iter` option allows you to specify the number of iterations to run for each set of initial values without fully fitting the model. The initial value set that shows the best performance during these test iterations is then used for the full estimation process to convergence. This approach helps ensure that the algorithm starts with a good initial value, thereby increasing the likelihood of finding the global maximum and improving overall model fitting.

Boundary solution problem

The boundary solution problem in LCA arises when estimated parameters reach the extreme limits of the parameter space, such as latent class probabilities or conditional response probabilities being estimated as exactly 0 or 1. This issue can occur due to several factors, including small sample sizes, which fail to provide enough data to accurately estimate parameters for all classes, and overly complex models with too many classes, which can cause the model to overfit the data and push some probabilities to extreme values. Boundary solutions lead to several problems. They can cause overfitting, where the model captures noise in the data instead of the underlying structure, reducing the model's generalizability. Furthermore, estimates at the boundary can be unstable and highly sensitive to small changes in the data, resulting in unreliable outcomes.

One effective approach to addressing boundary solutions is to fix parameters at boundary values, such as 0 or 1, and exclude them from estimation. By fixing these parameters, we can simplify the model and enhance the interpretability of the parameters. This method also improves the stability of the estimation process, as fixed boundary values reduce the sensitivity of the estimates to small data variations. In the **slca** package, we can fix specific parameters to zero, by designating the index of the parameters in `fix2zero` argument. The index of the parameters can be obtained by `param()` function with `index` being `TRUE`. We will observe how this works in following applications.

3.5. Model diagnostics

After estimating a structural latent class model, a critical subsequent step is to evaluate its adequacy and interpretability. Model diagnostics in the context of LCA serve two primary purposes: first, to determine the optimal number of latent classes that best represents the underlying heterogeneity in the data, and second, to assess how well the chosen model fits the observed data. A model that fits the data well should be able to accurately reproduce the relationships observed among the manifest variables. The **slca** package provides several tools to facilitate this evaluation process, focusing on goodness-of-fit statistics and robust inferential methods that are essential for making sound scientific conclusions.

Goodness of fit

Assessing the goodness of fit of a latent class model involves a comparison of a set of candidate models, which typically differ in the number of latent classes. This comparison is guided by various statistical indices that aim to measure the balance between model fit and parsimony.

A fundamental measure of absolute model fit is the likelihood-ratio chi-squared statistic, often denoted as G^2. This statistic provides an overall test of the model by comparing the observed frequencies of all possible response patterns in the manifest variables with the expected frequencies predicted by the estimated model. The G^2 statistic is calculated as

$$G^2 = 2 \sum_{j=1}^{J} f_j \log \left(\frac{f_j}{e_j} \right).$$

Here, J represents the total number of cells in the multidimensional contingency table formed by the manifest variables, f_j is the observed frequency (the actual count of individuals) for a specific response pattern j, and e_j is the expected frequency for that same pattern as predicted by the fitted model's parameters. The degrees of freedom for this test are calculated as $df = J - p - 1$, where p is the number of freely estimated parameters in the model. A non-significant p-value for the G^2 test suggests that the model-predicted frequencies are not significantly different from the observed frequencies, implying that the model adequately reproduces the observed data structure.

However, the chi-squared approximation for the distribution of G^2 is only reliable under specific conditions, namely when the expected frequencies in all cells of the contingency table are sufficiently large. In many practical LCA applications, especially those with several polytomous manifest variables, the number of possible response patterns J can be much larger than the sample size n. This leads to a very large and sparse contingency table, where most cells have observed frequencies of zero or one. In such common scenarios, the distribution of the G^2 statistic no longer follows a theoretical chi-squared distribution, rendering the associated p-value untrustworthy for assessing model fit (Collins and Lanza 2009).

To navigate this issue and to formally compare competing models (e.g., a 3-class versus a 4-class model), researchers heavily rely on relative fit indices, particularly information criteria. The **slca** package reports two of the most widely used criteria: the Akaike Information Criterion (AIC) and the Bayesian Information Criterion (BIC). Their formulas are $\text{AIC} = -2\log(L) + 2p$ and $\text{BIC} = -2\log(L) + p\log(n)$, respectively. Here, $\log(L)$ is the maximized log-likelihood of the model, p is the number of free parameters, and n is the sample size. Both AIC and BIC reward better fit (higher log-likelihood) but penalize complexity (more parameters). The BIC imposes a stronger penalty for complexity, especially in larger samples, and is generally favored in simulation studies for its superior performance in correctly identifying the true number of latent classes (Nylund *et al.* 2007). When comparing a set of models, the one with the lowest AIC or BIC value is considered to offer the best balance of fit and parsimony.

Beyond overall model fit, it is also crucial to evaluate the clarity of the latent class solution. The quality of classification, or how well individuals are distinguished by the latent classes, can be summarized by entropy. This index is calculated from the posterior probabilities of class membership for each individual. If the model provides a clear classification, most individuals will have a high posterior probability of belonging to one class and very low probabilities for all other classes. Conversely, poor classification is indicated by individuals having similar posterior probabilities for multiple classes. Entropy consolidates this information into a single value ranging from 0 to 1, where values closer to 1 signify a clearer class structure with less classification uncertainty.

Bootstrap p-value

To overcome the limitations of the asymptotic G^2 test, a more reliable and now standard approach for assessing model fit in LCA is the use of bootstrap-based procedures (Nylund *et al.* 2007). While information criteria such as AIC and BIC are invaluable for comparing a set of candidate models, they do not provide a significance test for a model's absolute fit. Furthermore, a conventional likelihood-ratio chi-squared difference test between models with different numbers of classes (e.g., k vs. $k+1$ classes) is inappropriate. This is because the null model (k classes) lies on the boundary of the parameter space of the alternative model ($k+1$ classes), which violates the regularity conditions required for the difference in log-likelihoods to follow a standard chi-squared distribution (Lo, Mendell, and Rubin 2001). Parametric bootstrap methods provide a robust solution to both of these challenges.

A key distinction must be made between two types of fit assessment. The first is the evaluation of *absolute model fit*, which assesses whether a single, specific model adequately explains the structure of the observed data. The second is the evaluation of *relative model fit*, which determines if a more complex model offers a statistically significant improvement over a simpler, more parsimonious competitor. The **slca** package supports both types of bootstrap-based assessments.

To evaluate the *absolute model fit*, we test whether a specific k-class model effectively explains the observed data. This procedure uses parametric bootstrap samples to generate an empirical null distribution for the G^2 statistic. The process begins by fitting the chosen model to the original data to obtain its parameter estimates and the observed G^2_{obs} value. Then, treating these estimates as the true population parameters, a large number of bootstrap samples (e.g., several hundred) are generated. The same k-class model is then fitted to each of these new samples, and a G^2 statistic is calculated for each fit. This collection of new G^2 statistics forms an empirical distribution, which represents the expected range of misfit if the proposed model were indeed the true data-generating process. The absolute fit of the original model is then assessed by locating its G^2_{obs} within this empirical distribution. The bootstrap p-value is the proportion of the bootstrap-generated G^2 values that are equal to or greater than the G^2_{obs}. A non-significant p-value suggests that the observed misfit is within the range of expected sampling error, and thus the model is considered to have good absolute fit.

To evaluate the *relative model fit*, one can directly compare two competing models, for example a k-class model versus a $(k+1)$-class model. This is achieved through the bootstrap likelihood ratio test (BLRT). This procedure tests the null hypothesis that the more parsimonious k-class model is sufficient. To conduct this test, parametric bootstrap samples are generated based on the parameter estimates of the simpler, k-class model. For each of these generated samples, both the k-class and the $(k+1)$-class models are estimated. From each pair of fits, the likelihood ratio test statistic, calculated as twice the difference in the log-likelihoods between the two models, is recorded. This process creates an empirical sampling distribution of the likelihood ratio statistic under the null hypothesis that the k-class model is true. Finally, the likelihood ratio statistic calculated from the original data is compared to this empirical distribution. The bootstrap p-value is the proportion of the bootstrap-generated statistics that exceed the one observed from the original data. A significant p-value indicates that the more complex $(k+1)$-class model provides a statistically significant improvement in fit, justifying the inclusion of an additional class.

3.6. Estimating covariate effects

As outlined in the previous chapter, **slca** uses a three-step methodology to estimate the effects of covariates. The standard or naive three-step approach begins by estimating a standard latent class model for a set of response variables (Y) without any covariates (step 1). From this initial model, posterior class membership probabilities, $P(X = x \mid Y_i)$, are calculated for each individual. Based on these probabilities, individuals are assigned to a class, W (step 2). This assignment can be done in two common ways: *modal* assignment, which assigns an individual to the single class with the highest posterior probability (a hard partitioning), or *proportional* assignment, which uses the posterior probabilities themselves as weights for each class (a soft partitioning). In the final step, the assigned class membership W is treated as an observed dependent variable and regressed on the covariates Z (step 3). This relationship is modeled using a multinomial logistic regression model. For identifiability, one class is treated as the baseline category, and its coefficients are fixed to zero. Assuming class K is the baseline, the model is

$$P(W = w \mid \mathbf{z}_i) = \frac{\exp(\mathbf{z}_i \boldsymbol{\beta}_w)}{\sum_{k=1}^{K} \exp(\mathbf{z}_i \boldsymbol{\beta}_k)} \text{ with } \boldsymbol{\beta}_K = \mathbf{0}.$$

The parameters are obtained by maximizing a weighted log-likelihood function:

$$\log L_{STEP3} = \sum_{i=1}^{n} \sum_{w=1}^{K} \omega_{iw} \log P(W = w \mid \mathbf{z}_i),$$

where under modal assignment, the weight ω_{iw} is 1 for the assigned class and 0 otherwise, and under proportional assignment, the weight $\omega_{iw} = P(X = w \mid \mathbf{y}_i)$.

However, this approach fails to account for classification uncertainty, which results in downward biased estimates for the covariate effects. To address this bias, there has been proposed two correction methods based on the relationship between the assigned class probabilities $P(W = w \mid \mathbf{z}_i)$ and the true class probabilities $P(X = x \mid \mathbf{z}_i)$:

$$P(W = w \mid \mathbf{z}_i) = \sum_{x=1}^{K} P(X = x \mid \mathbf{z}_i) P(W = w \mid X = x)$$

The BCH (Bolck, Croon, and Hagenaars) approach (Bolck, Croon, and Hagenaars 2004) corrects for bias by re-weighting data and maximizing a pseudo log-likelihood function:

$$\log L_{BCH} = \sum_{i=1}^{n} \sum_{x=1}^{K} \omega_{ix}^{*} \log P(X = x \mid \mathbf{z}_i),$$

where the true relationship between the covariates and the latent variable X is also modeled using a multinomial logistic function with baseline-category constraints:

$$P(X = x \mid \mathbf{z}_i) = \frac{\exp(\mathbf{z}_i \boldsymbol{\beta}_x)}{\sum_{k=1}^{K} \exp(\mathbf{z}_i \boldsymbol{\beta}_k)} \text{ with } \boldsymbol{\beta}_K = \mathbf{0}$$

The new weight ω_{ix}^{*} is obtained using the elements of the inverse of the classification error matrix, where the classification error is defined as $P(W = w \mid X = x)$.

Alternatively, the ML approach (Vermunt 2010) maximizes the following LCA-like log-likelihood function, which incorporates the classification errors, $P(W = w \mid X = x)$ as a measurement

error of LCA model:

$$\log L_{ML} = \sum_{i=1}^{n} \sum_{w=1}^{K} \omega_{iw} \log \left[\sum_{x=1}^{K} P(X = x \mid \mathbf{z}_i) P(W = w \mid X = x) \right]$$

Standard errors for the estimated regression coefficients are derived from the variance-covariance matrix, which is obtained by inverting the observed information matrix. For both the ML and BCH methods, this is calculated as the negative of the Hessian matrix of the respective log-likelihood function ($\log L_{ML}$ or $\log L_{BCH}$), evaluated at the final parameter estimates.

4. Package usage

4.1. Syntax for model construction: `slca()`

The model specification with `slca()` function is the starting point of the analysis.

```
slca(x, ..., constraints = NULL)
```

This primary function facilitates the construction of the latent structure of your analysis. We can input multiple formulas into the function to specify the desired model. The syntax of the formulas is organized into two main segments: *measurement model* specification and the *structural model* specification.

Measurement model specification

The latent class variables in the model are inferred from manifest indicators (observable variables), so it is necessary to specify the corresponding manifest indicators for each latent class variable. Each latent class variable is represented alongside its number of classes, denoted in either parentheses or brackets. For example, you might define a latent class variable with three classes, named `L1`, as `L1(3)` or `L1[3]`. The following syntax describes three latent class variables (`L1`, `L2`, `L3`), each with three classes, measured by their respective manifest indicators: `x`, `y`, and `z`.

```
L1(3) ~ x1 + x2 + x3
L2[3] ~ y1 + y2 + y3
L3(3) ~ z1 + z2 + z3
```

Each manifest indicator should be uniquely associated with its specific latent class variable, ensuring that the measurements are distinct and accurately reflect the underlying structure of the data.

Structural model specification

This segment focuses on establishing relationships between different latent class variables. For instance, `L1 ~ L2` indicates a relationship where latent class variable `L1` influences or is related to another latent class variable `L2`.

In some cases, it is necessary to define higher-level latent class variables that influence or consist of several other latent class variables. For example, a higher-level latent variable `P`

could be a composite of other latent variables `L1`, `L2`, and `L3`. Here, it is important to define the number of latent classes for `P`. This relationship can be represented as follows:

```
P[4] ~ L1 + L2 + L3
```

Similar to the measurement model, a latent class variables depend only on a single parent variable and cannot be influenced by other latent class variables. However, a latent class variable can influence multiple latent variables, acting as a parent to several children.

Parameter constraints

Ensuring measurement invariance is a critical aspect of model specification. Measurement invariance means that the same latent construct is measured consistently across different groups or time points. In **slca**, you can ensure measurement invariance by setting constraints on your latent class variables. The `constraints` argument specifies which latent class variables should be measured consistently. This feature is crucial for comparative studies, where it is essential to ensure consistent measurement properties of constructs across different sub-groups or over time.

```
R> slca(L1[3] ~ x1 + y1 + z1,
+       L2[3] ~ x2 + y2 + z2,
+       L3[3] ~ x3 + y3 + z3,
+       P[3] ~ L1 + L2 + L3,
+       constraints = c("L1", "L2", "L3"))
```

If the model needs to constrain the transition probabilities to be homogeneous in latent transition analysis, you can use `~` or `->` to represent the relationships, as shown below:

```
R> slca(L1[3] ~ x1 + y1 + z1,
+       L2[3] ~ x2 + y2 + z2,
+       L3[3] ~ x3 + y3 + z3,
+       L1 ~ L2, L2 ~ L3,
+       constraints = c("L1 ~ L2", "L2 -> L3"))
```

In this example, the transition probabilities from `L1` to `L2` and from `L2` to `L3` are constrained to be equal, ensuring that the transitions between latent class variables remain consistent.

4.2. Examples for existing models

Latent transition analysis

For the latent transition analysis (LTA), as specified in Equation 2 we will observe two example models using the function `slca()`. The first model involves three latent class variables: `L1`, `L2`, and `L3`. Each of these variables is measured by a distinct set of manifest variables at different time points. `L1` is measured by `x1`, `y1`, and `z1`; `L2` is measured by `x2`, `y2`, and `z2`; and `L3` is measured by `x3`, `y3`, and `z3`. The model encapsulates a sequential relationship among the latent class variables, where `L1` influences `L2`, and subsequently, `L2` influences `L3`. A key feature of this model is the assumption of measurement invariance across all latent class

variables, signified by the `constraints` applied uniformly to `L1`, `L2`, and `L3`. This assumption is critical as the same variables are measured at different time points, necessitating that the latent class variables maintain consistent meanings across these three time points.

```
R> lta1 <- slca(L1[3] ~ x1 + y1 + z1,
+               L2[3] ~ x2 + y2 + z2,
+               L3[3] ~ x3 + y3 + z3,
+               L1 ~ L2, L2 ~ L3,
+               constraints = c("L1", "L2", "L3"))
```

The second model presents a simpler structure with two latent class variables, `LG` and `LC`. Here, `LG` is measured by a set of `x` variables and `LC` by a set of `y` variables. This model delineates a direct relationship, where `LG` influences `LC` without assuming measurement invariance between the two variables. In this model, measurement invariance is not required because the two latent class variables are measured by different sets of manifest variables, indicating that they represent distinct constructs.

```
R> lta2 <- slca(LG[2] ~ x1 + x2 + x3,
+               LC[3] ~ y1 + y2 + y3 + y4,
+               LG ~ LC)
```

When the object is called, each model prints the information of model structure as follows:

```
R> lta1

Structural Latent Variable Model

Latent variables (Root*):
 Label: L1* L2 L3
nclass: 3   3  3

Measurement model:
 L1 -> { x1, y1, z1 }  a
 L2 -> { x2, y2, z2 }  a
 L3 -> { x3, y3, z3 }  a

Structural model:
 L1 -> { L2 }
 L2 -> { L3 }

Dependency constraints:
 A        B
 L1 -> L2 L2 -> L3

R> lta2

Structural Latent Variable Model
```

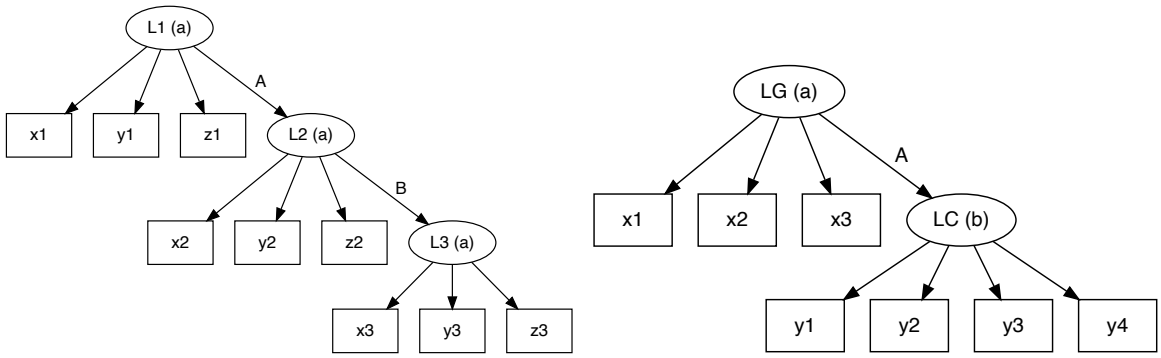

Figure 1: The graphical models for `lta1` (left) and `lta2` (right).

```
Latent variables (Root*):
 Label: LG* LC
nclass: 2   3

Measurement model:
 LG -> { x1, x2, x3 }      a
 LC -> { y1, y2, y3, y4 }  b

Structural model:
 LG -> { LC }

Dependency constraints:
 A
 LG -> LC
```

In the output of the first model `lta1`, the structural relationships among the latent class variables, as specified in the code, are evident. The structural model part clearly shows the sequential relationship, where `L1` influences `L2` and `L2` in turn influences `L3`. The measurement model section confirms the measurement constraints applied across the latent variables. The consistent label `a` across `L1`, `L2`, and `L3` indicates that the measurement probabilities for these latent class variables are constrained to be equal, reflecting the assumption of measurement invariance in the model. Among the latent class variables incorporated in the model, `L1` is not influenced by any other variable, so it is called *root* variable, denoted by asterisk (`*`).

Turning to the output of the second model `lta2`, the structural model portion succinctly presents a direct relationship between `LG` and `LC`, aligning with the simpler structure we designed. In the measurement model in `lta2`, the distinct labels `a` for `LG` and `b` for `LC` highlight the lack of measurement invariance between these two latent class variables, reflecting their representation of different aspects of the observation. We can plot the graphical models of each object by `plot()` function as follows:

```
R> plot(lta1)
R> plot(lta2)
```

In Figure 1, ellipses represent latent class variables, and rectangles represent observed vari-

ables. Latent class variables measured directly from observed variables are marked with lowercase letters in parentheses, indicating the corresponding measurement model. The edges between latent class variables are labeled with uppercase letters, representing the corresponding structural dependency. Note that the left model (`lta1`) in Figure 1 depicts changes in repeatedly measured latent class variables over time, while the right model (`lta2`) represents the relationships between different latent class variables.

The syntax for `lta1` defines a sequential structure representing change over time. The variable `L1` is implicitly defined as the root variable (i.e., `L1` does not appear on the right-hand side of a structural formula). The formulas `L1 ~ L2` and `L2 ~ L3` then establish the parent-child relationships, specifying that `L1` is the parent of `L2`, and `L2` is the parent of `L3`. This syntax directly implements the sequential transition structure of an LTA model.

Joint latent class analysis and latent class profile analysis

Under the hierarchical specification given in Equation 3, we explore two LCA model examples: joint latent class analysis (JLCA) and latent class profile analysis (LCPA). In the following example of JLCA, the model includes three latent class variables (`L1`, `L2`, and `L3`), each measured by a unique set of manifest variables (`x`, `y`, and `z`). Additionally, a fourth latent class variable (`JC`) is introduced and measured by the other three latent class variables, `L1`, `L2`, and `L3`. Here, `JC` is referred to as *joint latent class variable*. We can construct the JLCA model using the following commands.

```
R> jlca <- slca(L1[3] ~ x1 + x2 + x3,
+               L2[2] ~ y1 + y2 + y3,
+               L3[4] ~ z1 + z2 + z3,
+               JC[3] ~ L1 + L2 + L3)
```

The LCPA model also involves three latent class variables (`L1`, `L2`, and `L3`) but differs in its assumption of measurement invariance across these variables. Here, `PF`, called as *latent class profile variable* in LCPA, serves as an overarching latent class variable presenting the sequential patterns of `L1`, `L2`, and `L3` over time, and those three latent class variables are measured under a homogeneous constraint as indicated by the `constraints` argument, ensuring a consistent measurement structure across them similar to those in `lta1`.

```
R> lcpa <- slca(L1[3] ~ x1 + y1 + z1,
+               L2[3] ~ x2 + y2 + z2,
+               L3[3] ~ x3 + y3 + z3,
+               PF[3] ~ L1 + L2 + L3,
+               constraints = c("L1", "L2", "L3"))

R> jlca

Structural Latent Variable Model

Latent variables (Root*):
 Label: L1 L2 L3 JC*
nclass: 3  2  4  3
```

```
Measurement model:
 L1 -> { x1, x2, x3 }  a
 L2 -> { y1, y2, y3 }  b
 L3 -> { z1, z2, z3 }  c

Structural model:
 JC -> { L1, L2, L3 }

Dependency constraints:
 A         B         C
 JC -> L1 JC -> L2 JC -> L3
```

R> lcpa

```
Structural Latent Variable Model

Latent variables (Root*):
 Label: L1 L2 L3 PF*
nclass: 3  3  3  3

Measurement model:
 L1 -> { x1, y1, z1 }  a
 L2 -> { x2, y2, z2 }  a
 L3 -> { x3, y3, z3 }  a

Structural model:
 PF -> { L1, L2, L3 }

Dependency constraints:
 A         B         C
 PF -> L1 PF -> L2 PF -> L3
```

In the outputs of these models, the structural and measurement models are depicted, aligning with the specifications in the code. While the `jlca` output shows distinct measurement models for each latent class variable, the `lcpa` output presents measurement invariance by uniform measurement constraints across all latent class variables. Both outputs highlight the hierarchical structure of the models, with the root variables (`JC` in `jlca` and `PF` in `lcpa`) influencing the respective latent class variables. Thus, both overarching variables summarize the lower-level latent class variables (`L1`, `L2`, and `L3`). However, they differ in that `JC` summarizes the multidimensional aspect of the observation, while `PF` summarizes the longitudinal pattern of a certain latent attribute.

The graphical configuration of `jlca` and `lcpa` can be displayed as Figure 2 by `plot(jlca)` and `plot(lcpa)`. For the above model (`jlca`), the measurement models (lowercase alphabet) vary across latent class variables, `L1`, `L2`, and `L3`, while the measurement models (lowercase alphabet) are identical for the below model (`lcpa`) because `L1`, `L2`, and `L3` are repeatedly measured homogeneous variables.

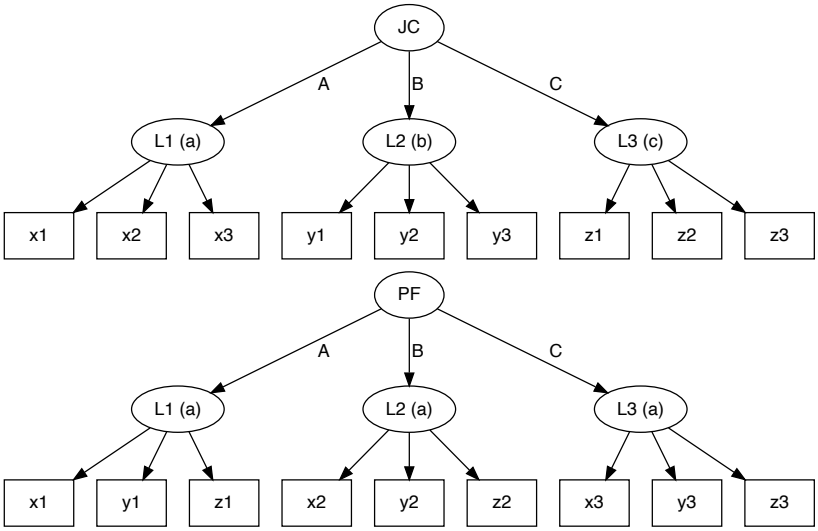

Figure 2: The graphical models for `jlca` (above) and `lcpa` (below).

The package syntax defines a layered structure for hierarchical models like JLCA and LCPA. In the `jlca` example, the formula `JC ~ L1 + L2 + L3` specifies a structure, where the higher-level variable `JC` is the single parent to each of the child variables `L1`, `L2`, and `L3`. This renders `JC` the root of the latent structure and implies that, conditional on `JC`, the child variables are mutually independent. The `lcpa` model is specified using identical syntax for the parent-child structure (`PF ~ L1 + L2 + L3`), but additionally uses the constraints argument to impose measurement invariance, reflecting that its child variables are treated as repeated measures of the same underlying construct.

More extensive latent class models

There are several extended models designed to investigate more complex latent structures as mentioned in Section 1. The additional latent class variables can influence or be influenced by joint latent class variable, and the joint latent class variable can be measured by different latent attributes. **slca** can construct this kind of complicated models as well. Following commands construct the model, where additional latent class variable affects the joint latent class variable.

```
R> model <- slca(L1[3] ~ x1 + x2 + x3,
+                L2[4] ~ y1 + y2 + y3,
+                L3[2] ~ z1 + z2 + z3,
+                G1[3] ~ g1 + g2 + g3,
+                JC[3] ~ L1 + L2 + L3,
+                G1 ~ JC)
R> plot(model)
```

Figure 3, generated by the `plot()` function, displays the graphical model. Similarly, we can extend the models by adding more latent class variables or additional layers. In Section 5.3, we will explore a more complex models such as joint latent class profile analysis or joint latent transition analysis using the real data example.

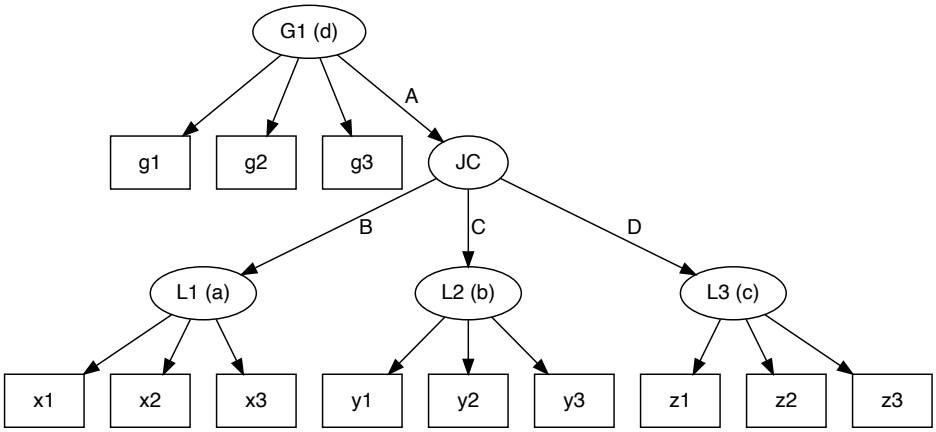

Figure 3: The graphical model for JLCA with an additional latent class variable.

4.3. Auxiliary functions

Function to estimate model

`estimate()` function is used to estimate the parameters of a model created with the `slca()` function. This function takes an `slca` object and a `data.frame` containing the observed variables to perform the estimation. We require `data.frame` in a wide format, where each row corresponds to an observation and each column to a manifest variable. The column names in the data frame must match the variable names specified in the model formula. **slca** is designed for categorical manifest variables, which can be provided as numeric vectors (with integers representing categories, e.g., 1, 2, 3) or as factors (ordered or unordered). Internally, all manifest variables are treated as nominal, and their levels are converted to integer codes starting from 1 for the analysis.

The function offers several estimation methods via the method argument:

`"em"`: The default method, which uses the expectation-maximization (EM) algorithm.

`"nlm"`: The Newton-Raphson algorithm via `nlm()` in base R.

`"hybrid"`: A combination approach that starts with the EM algorithm and refines the estimates using `nlm()`.

Users can also constrain specific parameters to zero using the `fix2zero` argument by providing a vector of parameter indices. The `param()` function with `index = TRUE` can be used to identify these indices. The estimation process can be further customized using the `control` argument, which accepts a list of parameters from the `slcaControl()` function.

The `slcaControl()` function allows users to specify and modify the control parameters for the estimation process in the estimate function. This provides fine-grained control over the fitting algorithms. Key arguments include:

`em.iterlim`: Sets the maximum number of iterations for the EM algorithm (default is `5000`).

`em.tol`: Defines the convergence tolerance for the EM algorithm (default is `1e-08`).

`nlm.iterlim`: Specifies the maximum iterations for the nlm function (default is `1000`).

`nlm.tol`: Sets the convergence tolerance for the `nlm` function (default is `1e-10`).

`nrep`: An integer for the number of estimation trials with different initial values (default is `1`).

`init.param`: A numeric vector to specify initial parameter values.

`verbose`: A logical value to indicate whether to display progress updates during estimation (default is `FALSE`).

`na.rm`: A logical value indicating whether to remove observations with missing values (default is `FALSE`).

It can be seamlessly integrated with the **magrittr** pipe operator (Bache and Wickham 2026) for a more streamlined workflow. For instance, the process of creating a model object with `slca()` and then passing it to the `estimate()` function can be written in two identical ways. The following example illustrates how a traditional nested function call is equivalent to the more readable, sequential workflow using the pipe operator.

```
R> model <- slca(status(3) ~ PAPRES + PADEG + MADEG)
R> estimate(model, data = gss7677)
R> slca(status(3) ~ PAPRES + PADEG + MADEG) %>%
+    estimate(data = gss7677)
```

Function to display estimated parameters

`param()` function is used to print the estimated parameters of a fitted `slcafit` object. It provides flexibility in how the parameters are displayed:

`type`: Specifies the format of the parameters, either as probabilities (`"probs"`, the default) or as log-odds (`"logit"`).

`se`: A logical argument that, when set to `TRUE`, displays the standard errors of the parameters instead of the estimates themselves (default is `FALSE`).

`index`: A logical argument that, when `TRUE`, includes the numerical indices of the estimated parameters in the output, which is useful for identifying parameters for constraints in other functions like `fix2zero` (default is `FALSE`).

The output is a list containing the estimated parameters, typically categorized into `PI` (membership probabilities for the root latent variable), `TAU` (conditional probabilities between latent variables), and `RHO` (item response probabilities).

Functions to diagnose and compare fitted models

Model diagnostics are crucial for assessing the adequacy and comparative effectiveness of estimated models, and this can be done using the `gof()` and `compare()` functions. The `gof()` (goodness-of-fit) function is designed to evaluate the absolute fit of estimated models. It computes several fit statistics, including the Akaike Information Criterion (AIC), the Bayesian Information Criterion (BIC), and the deviance statistic (G^2). The function can also perform statistical tests to evaluate absolute model fit via the `test` argument:

`"chisq"`: Conducts a chi-squared test based on the deviance statistic.

`"boot"`: Performs a more robust bootstrap goodness-of-fit test, which is computationally more intensive but offers a nuanced evaluation. The number of bootstrap samples can be specified with `nbooot` (default is `100`).

The `compare()` function extends these diagnostic capabilities by enabling the comparison of two fitted models to determine which provides a better relative fit to the data. This is essential when deciding between different model structures. Similar to `gof()`, `compare()` uses the deviance statistic (G^2) and allows for `"chisq"` or `"boot"` tests for the comparison. Together, these functions provide comprehensive diagnostic tools, allowing researchers not only to assess the adequacy of a single model but also to compare different models, ensuring the selection of the most appropriate model for their data.

Function to evaluate external covariate effect

The `regress()` function is designed to examine the influence of external variables on the latent class variables within a fitted `slca` model. The package implements a three-step approach to avoid the complexities and potential instabilities of a one-step approach, where covariates are included directly in the model estimation. This method performs a logistic regression to assess the effects of covariates on latent class membership. The function uses the last class as the baseline (reference) category for computing the odds ratios.

To address the potential for bias that can arise in a naive three-step approach, **slca** incorporates bias-correction methods. The method argument allows the user to choose the approach:

`"naive"`: A simple approach with no correction for classification error.

`"BCH"`: The bias-adjusted method proposed by Bolck *et al.* (2004).

`"ML"`: A maximum likelihood approach that accounts for classification error proposed by Vermunt (2010).

The `formula` argument specifies the regression model (e.g., `LV ~ COV1 + COV2`), and the data argument provides the `data.frame` containing the exogenous variables. The function returns a `list` containing the regression coefficients, odds ratios, standard errors, and the variance-covariance matrix. The standard errors are derived by numerically calculated Hessian matrix from `nlm` function.

5. Examples

In this section, we demonstrate how to implement structural LCA with **slca**. We utilize the **tidyverse** package to take advantage of the pipeline operator, enhancing code readability and efficiency. To identify the latent structure and determine the number of latent classes for each latent variable, it is essential to construct and estimate the model first, followed by conducting various diagnostics for model selection. Additionally, several techniques are needed to refine the fitted model and enhance the clarity of the results. These techniques include reordering latent classes or fixing certain free parameters to zero to improve model interpretation and identifiability. Using the `gss7677`, `addhealth`, and `nlsy97` datasets included in **slca**, we will explore the application of structural LCA on real data examples.

```
R> library("slca")
R> library("tidyverse")
```

5.1. Relationship between two latent class variables: `gss7677`

The General Social Survey (GSS) dataset presents a valuable repository of sociopolitical attitudes in the United States, offering a unique opportunity to apply LCA for understanding complex social phenomena (National Opinion Research Center 1992a,b). While the GSS dataset encompasses a wide range of variables, our analysis primarily focuses on *tolerance towards minorities*. The LCA based on these manifest items are conducted following the methodologies described in McCutcheon (1985), Bakk and Kuha (2017), and Bakk and Kuha (2020).

By analyzing the GSS 1976-77 data using the **slca** package, we aim to uncover latent patterns and relationships within these critical social dimensions. This exploration serves as an example of how LCA can be effectively applied to real-world data to derive meaningful conclusions about societal trends and attitudes.

In this process, we will explain how the **slca** package can be used to determine the optimal number of latent classes and measure the effects of covariates. We begin by exploring the latent structure of attitudes towards various minority groups with respondents' tolerance based on their willingness to extend civil liberties to five specific groups: racists, communists, homosexuals, atheists, and militarists. Tolerance is determined through responses to three questions regarding each group's right to speak publicly, teach, and have their writings available in public libraries. If respondents agree to all three questions, they are considered tolerant of that specific minority group. Conversely, if they disagree with any one of the questions, they are classified as intolerant of that minority group. Those variables are prepared in dataset `gss7677` named `TOLRAC` (racists), `TOLCOM` (communists), `TOLHOMO` (homosexuals), `TOLATH` (athesits), and `TOLMIL` (militarists), respectively. Our objective latent class variable describing tolerance of minority is named as `TOL`.

Model construction and selection

We start with a simpler model (`m2`) and gradually increase complexity (`m3`, `m4`, `m5`), assessing each model's fit to determine the most appropriate representation of the underlying attitudes.

```
R> m2 <- slca(TOL(2) ~ TOLRAC + TOLCOM + TOLHOMO + TOLATH + TOLMIL) %>%
+    estimate(data = gss7677)
R> m3 <- slca(TOL(3) ~ TOLRAC + TOLCOM + TOLHOMO + TOLATH + TOLMIL) %>%
+    estimate(data = gss7677)
R> m4 <- slca(TOL(4) ~ TOLRAC + TOLCOM + TOLHOMO + TOLATH + TOLMIL) %>%
+    estimate(data = gss7677)
R> m5 <- slca(TOL(5) ~ TOLRAC + TOLCOM + TOLHOMO + TOLATH + TOLMIL) %>%
+    estimate(data = gss7677, control = list(em.iterlim = 10000))
```

By inputting the constructed '`slca`' model into the `estimate()` function, we can estimate the model parameters. Remark that the `control` argument manipulates the estimation procedure by adjusting the maximum number of iterations, convergence tolerance, number of initial values, and the number of test iterations. Also, we can choose whether to remove the

observations including any NA and whether to provide output or messages generated by the function during its execution by adjusting control argument (see slcaControl()). Using the gof() function, we assess the model fit as we move from simpler to more complex models.

```
R> gof(m2, m3, m4, m5, test = "boot")

Analysis of Goodness of Fit Table

   Df  logLik   AIC   BIC     Gsq Res. Df Pr(Boot)
m2 11 -6856.9 13736 13802 251.808      20   <2e-16 ***
m3 17 -6760.3 13554 13656  58.485      14   <2e-16 ***
m4 23 -6734.2 13514 13652   6.435       8     0.72
m5 29 -6732.5 13523 13697   2.948       2     0.90
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The above output includes various metrics such as log-likelihood, Akaike information criterion (AIC), Bayesian information criterion (BIC), and G^2 statistic. In this analysis, as we progress from model m2 to m5, the residual degrees of freedom decrease, indicating an increased number of parameters in the more complex models. The log-likelihood values, which measure the probability of the model given the data, improve from model m2 to m5, and the G^2 statistics, a measure of model fit, decrease from m2 to m5, suggesting improved fit with increased complexity. The bootstrap p-values using G^2 statistics, particularly notable in assessing the goodness of fit for latent class models, reveal that while m2 and m3 have a significantly poor fit, both m4 and m5 fit the data reasonably well. Considering AIC and BIC, which penalize for complexity, model m4 shows the lowest values, indicating it might offer the best balance between fitting the data well and not being overly complex.

We can reinforce these findings by using the compare() function with models m4 and m5. Using the compare() function, we can determine whether the additional complexity of m5 significantly improves the model fit over m4. The command of compare() and the output is as follows:

```
R> compare(m4, m5, test = "boot")

Analysis of Relative Model Fit

Model H0: m4
Model H1: m5
   Df  logLik   AIC   BIC    Gsq Res. Df Pr(Boot)
m4 23 -6734.2 13514 13652
m5 29 -6732.5 13523 13697 3.4867       6     0.36
```

The output displays the bootstrap p-value of 0.36, indicating that the additional complexity of the five-class model (m5) does not provide a significant advantage over the four-class model (m4) in terms of model fit. The comparison suggests that m4 is a more parsimonious and suitable model for explaining the data.

```
R> param(m4)


PI :
(TOL)
  class
        1       2       3       4
   0.2129  0.0989  0.1171  0.5711

RHO :
(a)
        class
response       1       2       3       4
   1(V1)  0.9138  0.0000  0.7495  0.0719
   2      0.0862  1.0000  0.2505  0.9281
   1(V2)  0.9447  0.6683  0.2534  0.0438
   2      0.0553  0.3317  0.7466  0.9562
   1(V3)  0.9552  0.7453  0.5621  0.1378
   2      0.0448  0.2547  0.4379  0.8622
   1(V4)  0.9812  0.4212  0.6031  0.0357
   2      0.0188  0.5788  0.3969  0.9643
   1(V5)  0.9143  0.3755  0.3912  0.0551
   2      0.0857  0.6245  0.6088  0.9449

    V1     V2     V3      V4     V5
TOL TOLRAC TOLCOM TOLHOMO TOLATH TOLMIL
```

The `param()` function allows us to print the estimated parameters for the model. For tolerance (`TOL`), the model delineates four distinct classes as shown in the `PI`-parameter. On the bottom of the `RHO`-parameter, you can notice which latent class membership and manifest indicators are relevant to those `RHO`-parameter set. Noting that tolerance for each minority group is coded as 1 (tolerant) and intolerance as 2 (intolerant), approximately 0.21 of the sample can be labeled as "Tolerant" (class 1). This group stands out with a high level of tolerance across all minority groups, showing over 0.9 tolerance for each group. This indicates a broad and inclusive perspective with a strong willingness to accept diverse viewpoints and backgrounds. Class 2, representing approximately 0.10 of the sample, can be labeled as "Intolerant for Right." This group displays low tolerance towards racists but high tolerance towards communists and homosexuals. This indicates a distinct bias against right-leaning ideologies while being more accepting of individuals with differing sexual orientations. Class 3, constituting around 0.12 of the sample, can be labeled as "Intolerant for Left." Members of this class exhibit moderate levels of tolerance, displaying tolerance towards racists and atheists but lower tolerance levels towards communists and homosexuals. Class 4 is the largest group, comprising 0.57 of the sample. Members of this class appear to have low levels of tolerance towards all minority groups, with tolerance rates ranging from 0.03 to 0.14. This suggests a general lack of acceptance or openness to various minority groups within this segment of the population.

Regression on latent class variable

There are several variables indicating social status, such as parental education levels (`PADEG`, `MADEG`) and the prestige of the father's occupation (`PAPRES`). It is possible to analyze these variables as covariates for the latent class variable, `TOL`, using the `regress` function. The education levels (i.e., `PADEG` and `MADEG`) are coded from 1 to 5, corresponding to responses (1) lower than high school, (2) high school, (3) college, (4) bachelor and (5) graduate while `PAPRES` is coded from 1 to 3, corresponding to responses (1) low, (2) medium, and (3) high. We can assess the effect of each covariate individually as follows:

```
R> m4 %>%
+    regress(TOL ~ PADEG, data = gss7677)

Coefficients:
class  (Intercept)  PADEGHIGH-SCH  PADEGCOLLEGE  PADEGBACHELOR  PADEGGRADUATE
  1/4    -1.419         1.295          1.573         2.326          2.236
  2/4    -2.078         0.891          0.287         1.340          1.213
  3/4    -2.156         0.327        -10.268         0.119          0.598

R> m4 %>%
+    regress(TOL ~ MADEG, data = gss7677)

Coefficients:
class  (Intercept)  MADEGHIGH-SCH  MADEGCOLLEGE  MADEGBACHELOR  MADEGGRADUATE
  1/4    -1.6068        1.5465         2.0485        2.3377         2.2998
  2/4    -2.1565        1.0491         1.3453        0.9778         0.0763
  3/4    -2.2034        0.7506       -11.8871        0.5549       -11.6619

R> m4 %>%
+    regress(TOL ~ PAPRES, data = gss7677)

Coefficients:
class  (Intercept)  PAPRESMIDIUM  PAPRESHIGH
  1/4   -1.1281        0.0991        1.3617
  2/4   -1.8707       -0.0657        0.6002
  3/4   -1.8563       -0.4495        0.2981
```

Note that it is important to account for potential biases in the estimates, as the **slca** package uses a three-step estimation procedure. To adjust for this bias, the `method` argument in the `regress` function can be specified with options such as `ml` or `bch`, which are bias-adjusted approaches proposed by Bakk and Kuha (2020).

The regression results indicate that higher parental education levels and higher occupational prestige generally increase the likelihood of belonging to the tolerant class (class 1) towards minorities. While these covariates also show positive associations with the class labeled "Intolerant for Right" (class 2), the effects are less pronounced. For the "Intolerant for Left" class (class 3), the relationships are more complex and inconsistent, suggesting that the influence of these covariates varies significantly across different forms of intolerance.

Although it is feasible to explore the effects of each covariate through logistic regression using the three-step method, interpretation becomes increasingly difficult when considering all three categorical variables simultaneously. This complexity arises because each variable is categorical and may interact in ways that are challenging to decipher. To address this issue and simplify the interpretation, we summarize these covariates into a single latent variable representing overall social status, `STATUS`. Following a similar approach to the tolerance variable, we estimate the `slca` model and determine that a three-class solution provides the best fit for the data.

```
R> m3 <- slca(STATUS(3) ~ PAPRES + PADEG + MADEG) %>%
+    estimate(data = gss7677)
R> param(m3)

PI :
(STATUS)
  class
        1       2       3
   0.0571  0.2606  0.6823

RHO :
(a)
        class
response       1       2       3
   1(V1)  0.0000  0.2256  0.3599
   2      0.3685  0.6888  0.6294
   3      0.6315  0.0856  0.0107
   1(V2)  0.0000  0.0961  0.9545
   2      0.0000  0.8431  0.0421
   3      0.0406  0.0048  0.0034
   4      0.4277  0.0561  0.0000
   5      0.5317  0.0000  0.0000
   1(V3)  0.1289  0.1678  0.8333
   2      0.4471  0.7455  0.1556
   3      0.0090  0.0280  0.0036
   4      0.3029  0.0503  0.0065
   5      0.1121  0.0084  0.0011

       V1     V2    V3
STATUS PAPRES PADEG MADEG
```

The smallest class, comprising about 0.06 of the sample, indicates a predominant group with likely moderate to high levels of education and occupational prestige, based on the parents' background. This group represents a higher social status within society. The second class, making up about 0.26 of the sample, possibly corresponds to individuals from a middle social status. These individuals have a mix of medium and high levels of parental education and occupation, reflecting a diverse yet not the highest socio-economic background. The largest class, consisting of around 0.68 of the sample, might represent a lower social status segment.

This group is likely characterized by the lowest levels of parental education and occupational prestige, indicating limited socio-economic advantages.

In this phase of our analysis, we aim to deepen our understanding of the complex interplay between social status and tolerance. We integrate these two latent class variables into a unified model to investigate how social status influences an individual's level of tolerance. By examining this relationship comprehensively, we can uncover deeper sociopolitical insights within the GSS dataset.

We construct our unified analysis using **slca** again. This model is designed to capture the intricate connections between `STATUS`, which represents social status, and `TOL`, which represents tolerance. Based on our prior analysis, we've determined that the optimal numbers of latent classes for each latent class variable are three and four, respectively. To bridge these two variable, we introduce an additional relationship: `STATUS ~ TOL`.

```
R> lta <- slca(STATUS(3) ~ PAPRES + PADEG + MADEG,
+              TOL(4) ~ TOLRAC + TOLCOM + TOLHOMO + TOLATH + TOLMIL,
+              STATUS ~ TOL) %>%
+    estimate(data = gss7677)
```

Subsequently, we can examine the estimated parameters of our unified model:

```
R> param(lta)

PI :
(STATUS)
  class
        1       2       3
   0.0612  0.3105  0.6283

TAU :
(A)
     parent
child       1       2       3
    1  0.5947  0.3925  0.0988
    2  0.1623  0.2437  0.1070
    3  0.0477  0.1099  0.0918
    4  0.1953  0.2539  0.7024

parent STATUS
child  TOL

[...]
```

The output excludes the `RHO` parameters since they remain consistent with our previous analysis. However, even though the meaning of the latent classes remains the same, the class labels may be switched. Therefore, you should examine the `RHO` parameters and use the `reorder()` function to properly align the latent classes for accurate interpretation. In this example, the class order is aligned with the previous analysis.

The `TAU` parameters offer intriguing insights into the association between social status and tolerance. Notably, individuals with higher social status, represented by the first column, tend to exhibit greater tolerance, aligning with class 1 for the level of tolerance. Conversely, the third column, characterized by lower social status, leans towards greater intolerance, aligning with class 4 for the level of tolerance.

5.2. Transitions of a repeatedly measured latent class variable: `addhealth`

This empirical example replicates the analysis presented in Collins and Lanza (2009, p. 201) using the `addhealth` dataset in **slca**, which is drawn from a sample of 2,061 adolescents who participated in the Add Health study (Harris and Udry 2018). These adolescents were in the 10th and 11th grades during the 1994–1995 school year (wave 1) and in the 11th and 12th grades during the 1995–1996 school year (wave 2). In this example, we explore the measurement invariance assumption, reordering latent class order and parameter restrictions.

The `addhealth` data includes responses to various questions about adolescent depression. Adolescents reported how often they experienced feelings of depression during the past week. The survey items consisted of four questions related to sadness ("I felt that I could not shake off the blues even with help from my family and friends," "I felt depressed," "I felt lonely," "I felt sad"), two questions related to feelings of being disliked ("People were unfriendly to me," "I felt that people disliked me"), and two questions related to feelings of failure ("I thought my life had been a failure," "I felt life was not worth living"). Responses were coded as "yes" if the student reported experiencing the feeling "sometimes," "often," "most of the time," or "always," and as "no" if they reported experiencing the feeling "never or rarely." These recoded responses were used to create variables in the `addhealth` dataset. These behaviors are represented by the variables `S1-4`, `D1-2`, and `F1-2`, respectively. For Wave 1, these variables have a `w1` suffix (e.g., `S1w1`), and for Wave 2, they have a `w2` suffix (e.g., `D2w2`).

Model construction and selection

To construct a latent transition model, we can use the following commands:

```
R> lta4 <- slca(
+    DEP1(4) ~ S1w1 + S2w1 + S3w1 + S4w1 + D1w1 + D2w1 + F1w1 + F2w1,
+    DEP2(4) ~ S1w2 + S2w2 + S3w2 + S4w2 + D1w2 + D2w2 + F1w2 + F2w2,
+    DEP1 ~ DEP2) %>%
+    estimate(data = addhealth)
R> lta5 <- slca(
+    DEP1(5) ~ S1w1 + S2w1 + S3w1 + S4w1 + D1w1 + D2w1 + F1w1 + F2w1,
+    DEP2(5) ~ S1w2 + S2w2 + S3w2 + S4w2 + D1w2 + D2w2 + F1w2 + F2w2,
+    DEP1 ~ DEP2) %>%
+    estimate(data = addhealth)
R> lta6 <- slca(
+    DEP1(6) ~ S1w1 + S2w1 + S3w1 + S4w1 + D1w1 + D2w1 + F1w1 + F2w1,
+    DEP2(6) ~ S1w2 + S2w2 + S3w2 + S4w2 + D1w2 + D2w2 + F1w2 + F2w2,
+    DEP1 ~ DEP2) %>%
+    estimate(data = addhealth)
```

By adjusting number of classes from four to six, we can estimate the models with each

corresponding number of latent classes and determine the optimal number of latent classes for adolescent depression using the `gof()` function. It is notable that we assume that the latent class variables (`DEP1` and `DEP2`) having same number of classes.

```
R> gof(lta4, lta5, lta6)

Analysis of Goodness of Fit Table

      Df logLik   AIC   BIC    Gsq Res. Df
lta4  79 -14647 29452 29896 5656.5   65456
lta5 104 -14514 29235 29821 5389.8   65431
lta6 131 -14450 29162 29900 5262.9   65404
```

Based on the output, we could choose the five-class model, as it has the lowest BIC, while the AIC continuously declines as larger model. The G^2 statistics are not appropriate for checking the model adequacy in this analysis due to the sparsity of the contingency table, where the number of cells (65,536) is much larger than the number of observations (2,061).

Next, we need to consider the measurement invariance assumption. Since `DEP1` and `DEP2` are measured by the same indicators at different time points, we can assume consistency in their meanings to interpret changes over time more succinctly. By using the `constraints` argument with `c("DEP1", "DEP2")`, we can enforce identical conditional response probabilities for these latent class variables. However, we need to validate this assumption by comparing two models: one without measurement invariance (`lta5`) and one with measurement invariance (`lta5inv`).

```
R> lta5inv <- slca(
+    DEP1(5) ~ S1w1 + S2w1 + S3w1 + S4w1 + D1w1 + D2w1 + F1w1 + F2w1,
+    DEP2(5) ~ S1w2 + S2w2 + S3w2 + S4w2 + D1w2 + D2w2 + F1w2 + F2w2,
+    DEP1 ~ DEP2, constraints = c("DEP1", "DEP2")) %>%
+    estimate(data = addhealth)
R> compare(lta5, lta5inv, test = "chisq")

Analysis of Relative Model Fit

Model H0: lta5inv
Model H1: lta5
         Df logLik   AIC   BIC    Gsq Res. Df Pr(>Chi)
lta5inv  64 -14535 29197 29558
lta5    104 -14514 29235 29821 42.332      40   0.3707
```

Using a chi-squared test with G^2 statistics to compare these two models, we assess the validity of the measurement invariance assumption. With a p-value of 0.3707, we can assume measurement invariance, as the model with this assumption adequately explains the data. In this case, we used chi-squared test since `lta5inv` is nested in `lta5`, in that the model with measurement invariance is formulated by constraining the parameters of the model without measurement invariance.

Next, we explore the fitted model with the estimated parameters.

```
R> param(lta5inv)
PI :
(DEP1)
  class
      1      2      3      4      5
 0.2308 0.3632 0.1128 0.1106 0.1826

TAU :
(A)
     parent
child      1      2      3      4      5
    1 0.5442 0.1242 0.3350 0.1576 0.0478
    2 0.3024 0.7543 0.0325 0.0808 0.2499
    3 0.0679 0.0206 0.3595 0.1408 0.1716
    4 0.0444 0.0152 0.1081 0.5911 0.0569
    5 0.0412 0.0857 0.1650 0.0297 0.4738

parent DEP1
child  DEP2

RHO :
(a)
        class
response      1      2      3      4      5
   1(V1) 0.5150 0.0342 0.6930 0.8118 0.1184
   2     0.4850 0.9658 0.3070 0.1882 0.8816
   1(V2) 0.6799 0.0568 0.8401 0.9393 0.1826
   2     0.3201 0.9432 0.1599 0.0607 0.8174
   1(V3) 0.5458 0.0640 0.8385 0.8472 0.2704
   2     0.4542 0.9360 0.1615 0.1528 0.7296
   1(V4) 0.7948 0.1067 0.9198 0.9423 0.3406
   2     0.2052 0.8933 0.0802 0.0577 0.6594
   1(V5) 0.1603 0.0912 0.7641 0.6491 0.6809
   2     0.8397 0.9088 0.2359 0.3509 0.3191
   1(V6) 0.1253 0.0270 0.8827 0.7926 0.6705
   2     0.8747 0.9730 0.1173 0.2074 0.3295
   1(V7) 0.1297 0.0136 0.1260 0.8885 0.0767
   2     0.8703 0.9864 0.8740 0.1115 0.9233
   1(V8) 0.0554 0.0034 0.0576 0.6961 0.0475
   2     0.9446 0.9966 0.9424 0.3039 0.9525

     V1   V2   V3   V4   V5   V6   V7   V8
DEP1 S1w1 S2w1 S3w1 S4w1 D1w1 D2w1 F1w1 F2w1
DEP2 S1w2 S2w2 S3w2 S4w2 D1w2 D2w2 F1w2 F2w2
```

By examining `RHO` parameters, we can interpret the meaning of each latent class of `DEP1` and `DEP2`. Since the response code 1 refers to "yes" (endorsed the feeling) and 2 refers to "no"

(did not endorse the feeling), we examine the probabilities of endorsing each item within each class to determine the characteristics of that class. Class 1, comprising approximately 0.23 of the sample, shows a high probability of endorsing sadness-related items (`V1`–`V4`) but low probabilities for other items. This class could be labeled as "Sad." The largest group, class 2, which makes up about 0.36 of the sample, represents individuals with very low probabilities of endorsing any depressive symptoms across all items, indicating minimal experience with them. This group could be labeled as "Not Depressed." Class 3, comprising approximately 0.11 of the sample, has high probabilities of endorsing both sadness (`V1`–`V4`) and feelings of being disliked, (`V5`, `V6`) but lower probabilities for feelings of failure (`V7`, `V8`). This class could be labeled as "Sad + Disliked." Class 4, also making up about 0.11 of the sample, shows high probabilities of endorsing all depressive symptoms, including feelings of failure. These individuals experience comprehensive depressive symptoms across all domains, so this class could be labeled as "Depressed." Finally, class 5, which consists of individuals who make up about 0.18 of the sample, exhibits high probabilities of endorsing items related to feelings of being disliked (`V5`, `V6`) but low probabilities for other items. This class could be labeled as "Disliked."

Class reordering

To better reflect the severity of depressive symptoms, we reordered the classes to arrange them from least to most severe: class 2 (Not Depressed), class 1 (Sad), class 5 (Disliked), class 3 (Sad + Disliked), and class 4 (Depressed). Using the `reorder()` function, we adjusted the class order accordingly.

```
R> lta5sort <- lta5inv %>%
+    reorder(DEP1 = c(2, 1, 5, 3, 4))
R> param(lta5sort)

PI :
(DEP1)
  class
        1       2       3       4       5
   0.3632  0.2308  0.1826  0.1128  0.1106

TAU :
(A)
     parent
child       1       2       3       4       5
    1  0.7543  0.3024  0.2499  0.0325  0.0808
    2  0.1242  0.5442  0.0478  0.3350  0.1576
    3  0.0857  0.0412  0.4738  0.1650  0.0297
    4  0.0206  0.0679  0.1716  0.3595  0.1408
    5  0.0152  0.0444  0.0569  0.1081  0.5911

parent DEP1
child  DEP2

[...]
```

The redundant `RHO` parameters are omitted since we can distinguish class order by `PI` parameters. Now, the reordered `TAU (A)` matrix indicates the likelihood of shifting from a column's latent status at time 1 (parent, `DEP1`) to a row's latent status at time 2 (child, `DEP2`). Diagonal elements represent the probability of remaining in the same status over time. Individuals in the "Not Depressed" (class 1) status at time 1 have a high probability (0.75) of remaining in the same status at time 2, indicating substantial stability. Over time, some individuals may transition to the "Sad" (class 2) status (0.12) or the "Disliked" (class 3) status (0.09), but the probability of worsening to more severe states is very low. Those in the "Sad" and "Disliked" status have a moderate probability (0.54, 0.47) of remaining in the same status and a significant probability of transitioning to the Not Depressed status (0.30 and 0.25, respectively), suggesting that feelings of sad or being disliked may diminish over time for some individuals. However, there is also a notable probability of moving from the "Disliked" to the "Sad + Disliked" (class 4) status (0.17), highlighting a risk of worsening symptoms, while the transition from the "Sad" to the "Sad + Disliked" is relatively small (0.07). Those in the "Sad + Disliked" status show moderate stability (0.36) but also significant probabilities of improving to the "Sad" (0.34) or to the "Disliked" (0.17). Individuals in the "Depressed" (class 5) status have a high probability (0.59) of remaining depressed, indicating that it is relatively difficult for adolescents with severe depressive symptoms to experience improvement over time.

Parameter restriction

Additionally, there might be a hypothesis that certain transitions between specific pairs of latent statuses are theoretically impossible. To test hypotheses about the nature or extent of change, researchers can impose restrictions on elements of the transition probability matrix and compare the fit of the restricted model to a less restricted one. If the simpler, more restricted model fits the data as well as the less restricted model, it is preferred due to its simplicity. Conversely, if the restricted model fits the data significantly worse, it suggests that the additional parameters in the less restricted model are necessary, and the hypothesis imposed by the restrictions may be rejected. This approach to hypothesis testing provides direct insight into scientific conclusions about change.

To illustrate how hypotheses about change can be expressed through restrictions on transition probabilities, consider a scenario where a researcher wants to test a specific hypothesis about the progression of depression. Suppose the latent statuses "Sad" and "Disliked" are believed to be distinctly different starting points in the development of depression. The hypothesis states that while it is possible for individuals to move from "Sad" to "Sad + Disliked" and from "Disliked" to "Sad + Disliked," it is not possible to transition directly from "Sad" to "Disliked" or from "Disliked" to "Sad." To test this hypothesis, certain transition probabilities corresponding to these impossible transitions need to be set to zero. To identify the specific parameters associated with these transitions, we can display the parameter indices by setting the argument `index = TRUE` when printing the model parameters. This allows us to see which parameters correspond to the transitions we wish to restrict.

```
R> lta5sort %>%
+    param(index = TRUE)

PI :
(DEP1)
```

```
  class
            1           2           3           4           5
    0.3632 (1)  0.2308 (2)  0.1826 (3)  0.1128 (4)  0.1106 (5)

TAU :
(A)
     parent
child            1            2            3            4            5
    1   0.7543 (6)  0.3024 (11)  0.2499 (16)  0.0325 (21)  0.0808 (26)
    2   0.1242 (7)  0.5442 (12)  0.0478 (17)  0.3350 (22)  0.1576 (27)
    3   0.0857 (8)  0.0412 (13)  0.4738 (18)  0.1650 (23)  0.0297 (28)
    4   0.0206 (9)  0.0679 (14)  0.1716 (19)  0.3595 (24)  0.1408 (29)
    5  0.0152 (10)  0.0444 (15)  0.0569 (20)  0.1081 (25)  0.5911 (30)

parent DEP1
child  DEP2

[...]
```

Once the relevant parameters are identified (in this case, parameters 13 and 17), we fix them to zero using the appropriate command and re-estimate the model. By doing so, we create a restricted model that embodies our hypothesis.

```
R> lta5fix <- lta5sort %>%
+    estimate(fix2zero = c(13, 17))
```

Finally, we compare the restricted model with the original, less restricted model to assess the impact of our constraints. This comparison is typically done using the chi-squared test, to determine if the restricted model provides a significantly worse fit to the data.

```
R> compare(lta5inv, lta5fix, test = "chisq")

Analysis of Relative Model Fit

Model H0: lta5fix
Model H1: lta5inv
        Df logLik   AIC   BIC    Gsq Res. Df Pr(>Chi)
lta5fix 62 -14538 29199 29548
lta5inv 64 -14535 29197 29558 5.7944       2  0.05518 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Based on the results of the hypothesis testing, we found that the restricted model did not fit the data significantly worse than the unrestricted model (p-value is 0.055). Therefore, we do not reject the hypothesis that direct transitions between "Sad" and "Disliked" are impossible. This suggests that "Sad" and "Disliked" represent distinctly different starting points in the development of depression among adolescents.

5.3. Multiple repeatedly measured latent class variables: `nlsy97`

The National Longitudinal Survey of Youth 1997 (NLSY97) provides a rich dataset tracking American adolescents as they transition from school to adulthood (Bureau of Labor Statistics, U.S. Department of Labor 2026). We focus on a subset of 1,004 individuals who began drinking before the age of 13. The dataset includes detailed responses over three key life stages—adolescence, young adulthood, and adulthood—covering their usage patterns of tobacco, alcohol, and marijuana. The survey's questions have been adapted into 12 binary items, offering a nuanced view of substance use behaviors at different stages.

Our goal is to replicate and extend the analysis conducted in Jeon, Seo, Anthony, and Chung (2020) using this dataset. The dataset is included in the **slca** package under the name `nlsy97`, and it has been meticulously prepared for this specific investigation. The variables in `nlsy97` are as follows:

```
R> names(nlsy97)
```

```
 [1] "SEX"     "RACE"    "ESMK_98" "FSMK_98" "DSMK_98" "HSMK_98"
 [7] "EDRK_98" "CDRK_98" "WDRK_98" "BDRK_98" "EMRJ_98" "CMRJ_98"
[13] "OMRJ_98" "SMRJ_98" "ESMK_03" "FSMK_03" "DSMK_03" "HSMK_03"
[19] "EDRK_03" "CDRK_03" "WDRK_03" "BDRK_03" "EMRJ_03" "CMRJ_03"
[25] "OMRJ_03" "SMRJ_03" "ESMK_08" "FSMK_08" "DSMK_08" "HSMK_08"
[31] "EDRK_08" "CDRK_08" "WDRK_08" "BDRK_08" "EMRJ_08" "CMRJ_08"
[37] "OMRJ_08" "SMRJ_08"
```

The first two, `SEX` and `RACE`, provide demographic information about the respondents. The remaining variables are binary indicators derived from the NLSY97 survey questions, pertaining to tobacco, alcohol, and marijuana use across three specific years: 1998, 2003, and 2008. The suffixes `_98`, `_03`, and `_08` in the variable names denote the year of the survey wave. For example, `ESMK_98` refers to 'Ever Smoked in 1998,' `FSMK_98` to 'Frequent Smoke in 1998,' and so on, with similar naming conventions for the years 2003 (`_03`) and 2008 (`_08`). These variables represent different aspects of substance use, such as ever engaging in substance use–ever smoked (`ESMK_`), ever drunk (`EDRK_`), and ever used marijuana (`EMRJ_`); frequency of use–frequent smoker (`FSMK_`), daily smoker (`DSMK_`), weekly drinker (`WDRK_`), and occasional marijuana user (`OMRJ_`); current use–current drinker (`CDRK_`) and current marijuana user (`CMRJ_`); and intensity of use–heavy smoker (`HSMK_`), binge drinker (`BDRK_`), and school/work marijuana user (`SMRJ_`). The detailed process of deriving these binary indicators from the original survey questions is thoroughly documented in Jeon *et al.* (2020).

Model construction

Using this dataset, our aim is to construct structural LC models, which have both longitudinal and multidimensional attributes. These analyses will incorporate longitudinal responses of various substance use behaviors. We plan to undertake joint latent class profile analysis conducted in Jeon *et al.* (2020), offering a comprehensive exploration of substance use patterns among youth as they transition from adolescence to adulthood using the **slca** package.

```
R> jlcpa <- slca(
+  # Smoking behavior at each time point
```

```
+    SMK_98(3) ~ ESMK_98 + FSMK_98 + DSMK_98 + HSMK_98,
+    SMK_03(3) ~ ESMK_03 + FSMK_03 + DSMK_03 + HSMK_03,
+    SMK_08(3) ~ ESMK_08 + FSMK_08 + DSMK_08 + HSMK_08,
+  # Drinking behavior at each time point
+    DRK_98(3) ~ EDRK_98 + CDRK_98 + WDRK_98 + BDRK_98,
+    DRK_03(3) ~ EDRK_03 + CDRK_03 + WDRK_03 + BDRK_03,
+    DRK_08(3) ~ EDRK_08 + CDRK_08 + WDRK_08 + BDRK_08,
+  # Marijuana use behavior at each time point
+    MRJ_98(3) ~ EMRJ_98 + CMRJ_98 + OMRJ_98 + SMRJ_98,
+    MRJ_03(3) ~ EMRJ_03 + CMRJ_03 + OMRJ_03 + SMRJ_03,
+    MRJ_08(3) ~ EMRJ_08 + CMRJ_08 + OMRJ_08 + SMRJ_08,
+  # Overall substance use behavior at each time point
+    SUB_98(5) ~ SMK_98 + DRK_98 + MRJ_98,
+    SUB_03(5) ~ SMK_03 + DRK_03 + MRJ_03,
+    SUB_08(5) ~ SMK_08 + DRK_08 + MRJ_08,
+  # Longitudinal shifts of overall substance use behavior
+    PROF(4) ~ SUB_98 + SUB_03 + SUB_08,
+  # Measurement invariance of latent class variables
+    constraints = list(
+       c("SMK_98", "SMK_03", "SMK_08"),
+       c("DRK_98", "DRK_03", "DRK_08"),
+       c("MRJ_98", "MRJ_03", "MRJ_08"),
+       c("SUB_98 ~ SMK_98", "SUB_03 ~ SMK_03", "SUB_08 ~ SMK_08"),
+       c("SUB_98 ~ DRK_98", "SUB_03 ~ DRK_03", "SUB_08 ~ DRK_08"),
+       c("SUB_98 ~ MRJ_98", "SUB_03 ~ MRJ_03", "SUB_08 ~ MRJ_08")
+    )
+ )
```

This provided code is meticulously crafted to analyze the `nlsy97` dataset by identifying the following latent class variables, focusing on the patterns of substance use across different life stages.

1. *Smoking behavior* (`SMK_98`, `SMK_03`, `SMK_08`):
   Smoking behavior is modeled for each survey wave (1998, 2003, and 2008) using indicators such as ever smoked, frequency of smoking, daily smoking, and heavy smoking. These behaviors are categorized into three latent classes, representing different levels of smoking intensity and frequency at each time point.

2. *Drinking behavior* (`DRK_98`, `DRK_03`, `DRK_08`):
   Drinking behavior is similarly analyzed at each time point with corresponding indicators like ever drunk, current drinker, weekly drinker, and binge drinker. The model categorizes these behaviors into three latent classes, illustrating varying drinking patterns over time.

3. *Marijuana use behavior* (`MRJ_98`, `MRJ_03`, `MRJ_08`):
   Marijuana use at each survey wave is also modeled using indicators like ever used marijuana, current marijuana user, and school/work marijuana user. This is again broken down into three latent classes to depict different levels of marijuana use.

Additionally, overarching latent class variables, SUB_98, SUB_03, and SUB_08 combines smoking, drinking, and marijuana use behaviors at each survey wave, categorizing them into five latent classes. This provides a comprehensive view of overall substance use patterns at each time point.

Lastly, higher-level latent class variable PROF is introduced to encapsulate the overall substance use profile across all time points (i.e., SUB_98, SUB_03, and SUB_08). The variable PROF is modeled to have four latent classes, summarizing the comprehensive substance use patterns across the three survey waves. This approach, known as joint latent class profile analysis (JLCPA), provides deeper insights into the overarching trends and profiles in substance use.

Instead of introducing a higher-level latent class variable PROF, we can investigate the longitudinal shift of the substance use behavior by modeling transition probabilities between SUB variables of each survey wave by adding command SUB_98 ~ SUB_03 and SUB_03 ~ SUB_08 instead of PROF(4) ~ SUB_98 + SUB_03 + SUB_08. The model examines transitions in substance use behavior over time, utilizing latent transition analysis (LTA) on joint class variables. This approach, known as joint latent transition analysis (JLTA), helps in understanding how substance use behaviors evolve from adolescence to young adulthood and into later adulthood sequentially by exploring transition probabilities.

To ensure consistency in the measurement of substance use behaviors across different survey waves, constraints are applied: Each substance use behavior and the overall substance use behavior are measured in a similar manner across different time points. The JLCPA model incorporates 13 latent class variables, including latent class variables for smoking, drinking, and marijuana use at each of the three time points (i.e., SMK_, DRK_, and MRJ_), the joint class variables at each time point capturing the overall substance use behavior (i.e, SUB_), and the joint class profile variable (i.e., PROF) encapsulating the longitudinal pattern of overall substance use, SUB_98, SUB_03, and SUB_08. The initial part of the model description for JLCPA is as follows:

```
R> jlcpa

Structural Latent Variable Model

Latent variables (Root*):
 Label: SMK_98 SMK_03 SMK_08 DRK_98 DRK_03 DRK_08 MRJ_98 MRJ_03 MRJ_08
nclass: 3      3      3      3      3      3      3      3      3

 Label: SUB_98 SUB_03 SUB_08 PROF*
nclass: 5      5      5      4

Measurement model:
 SMK_98 -> { ESMK_98, FSMK_98, DSMK_98, HSMK_98 }  a
 SMK_03 -> { ESMK_03, FSMK_03, DSMK_03, HSMK_03 }  a
 SMK_08 -> { ESMK_08, FSMK_08, DSMK_08, HSMK_08 }  a
 DRK_98 -> { EDRK_98, CDRK_98, WDRK_98, BDRK_98 }  b
 DRK_03 -> { EDRK_03, CDRK_03, WDRK_03, BDRK_03 }  b
 DRK_08 -> { EDRK_08, CDRK_08, WDRK_08, BDRK_08 }  b
```

```
 MRJ_98 -> { EMRJ_98, CMRJ_98, OMRJ_98, SMRJ_98 }  c
 MRJ_03 -> { EMRJ_03, CMRJ_03, OMRJ_03, SMRJ_03 }  c
 MRJ_08 -> { EMRJ_08, CMRJ_08, OMRJ_08, SMRJ_08 }  c

Structural model:
 SUB_98 -> { SMK_98, DRK_98, MRJ_98 }
 SUB_03 -> { SMK_03, DRK_03, MRJ_03 }
 SUB_08 -> { SMK_08, DRK_08, MRJ_08 }
 PROF   -> { SUB_98, SUB_03, SUB_08 }

Dependency constraints:
 A               B               C
 PROF -> SUB_98 PROF -> SUB_03 PROF -> SUB_08
 D                 E                 F
 SUB_98 -> SMK_98 SUB_98 -> DRK_98 SUB_98 -> MRJ_98
 SUB_03 -> SMK_03 SUB_03 -> DRK_03 SUB_03 -> MRJ_03
 SUB_08 -> SMK_08 SUB_08 -> DRK_08 SUB_08 -> MRJ_08
```

The measurement model is consistent across time points, indicated by labels `a`, `b`, and `c`, ensures a uniform approach to measuring each type of substance use behavior.

In the structural model of JLCPA, each of joint latent class variables (i.e., `SUB_98`, `SUB_03`, `SUB_08`) impacts only the indirectly measured variables corresponding to distinct substances, rather than influencing the subsequent joint latent class variable. Instead, the latent class profile variable `PROF` exerts influence on these joint latent class variables, offering a holistic view of substance use across the entire study period.

Lastly, the dependency constraints part describes the measurement invariance among transition probabilities. This part of the output indicates that the transition probabilities from the overall substance use behavior (`SUB_`) to the specific behaviors of smoking (`SMK_`), drinking (`DRK_`), and marijuana use (`MRJ_`) are maintained consistently across the different time points. In this case, the constraints `D`, `E`, and `F` represent the constrained influence of `SUB_` on each specific substance use behavior, ensuring a consistent measurement approach across time points. The `A`, `B`, and `C` reveal the influence of the profile variable `PROF` on the `SUB` variables at each time point. This aspect of the model allows for an analysis of the summarized longitudinal shifts in overall substance use behavior, providing a broader perspective on how patterns of substance use evolve over time within the cohort.

The model configurations of JLCPA for `nlsy97` can be displayed by `plot()` function and shown in Figure 4. Note that the corresponding measurement model and structural dependency can be identified by the lower-case alphabet in the parenthesis and upper-case alphabet on the edges, respectively.

Model inference

By utilizing the `estimate()` function on the JLCPA model, we can estimate its parameters effectively. The extraction of these parameter estimates is subsequently conducted using the `param()` function as follows:

```
R> jlcpa_fit <- estimate(jlcpa, data = nlsy97)
```

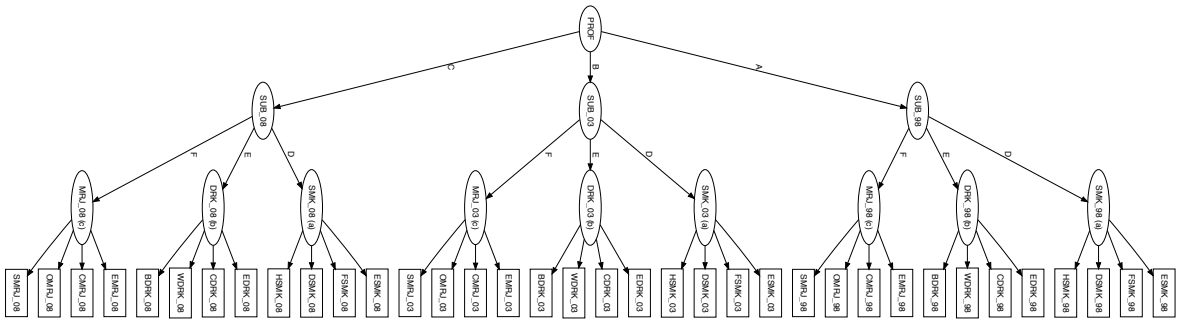

Figure 4: The graphical models of JLCPA for the `nlsy97` dataset.

```
R> param(jlcpa_fit)
```

From the estimated parameters, researchers can label the latent classes in a meaningful way. This labeling process focuses on `SMK`, `DRK`, `MRJ`, and then extends to `SUB`. We start with the `RHO` section of the parameter estimates. Following is the `RHO` section of the output:

```
RHO :
(a)
        class
response       1       2       3
   1(V1)  0.0134  1.0000  1.0000
   2      0.9866  0.0000  0.0000
   1(V2)  0.0000  0.6701  1.0000
   2      1.0000  0.3299  0.0000
   1(V3)  0.0000  0.0000  0.8833
   2      1.0000  1.0000  0.1167
   1(V4)  0.0000  0.0000  0.6460
   2      1.0000  1.0000  0.3540

       V1      V2      V3      V4
SMK_98 ESMK_98 FSMK_98 DSMK_98 HSMK_98
SMK_03 ESMK_03 FSMK_03 DSMK_03 HSMK_03
SMK_08 ESMK_08 FSMK_08 DSMK_08 HSMK_08
(b)
        class
response       1       2       3
   1(V1)  0.0759  1.0000  1.0000
   2      0.9241  0.0000  0.0000
   1(V2)  0.0000  0.5760  1.0000
   2      1.0000  0.4240  0.0000
   1(V3)  0.0000  0.0000  0.6934
   2      1.0000  1.0000  0.3066
   1(V4)  0.0000  0.0000  0.8061
   2      1.0000  1.0000  0.1939
```

```
       V1      V2      V3      V4
DRK_98 EDRK_98 CDRK_98 WDRK_98 BDRK_98
DRK_03 EDRK_03 CDRK_03 WDRK_03 BDRK_03
DRK_08 EDRK_08 CDRK_08 WDRK_08 BDRK_08
(c)
        class
response       1       2       3
   1(V1)  0.0002  0.5189  1.0000
   2      0.9998  0.4811  0.0000
   1(V2)  0.0000  0.0000  1.0000
   2      1.0000  1.0000  0.0000
   1(V3)  0.0000  0.0000  0.5259
   2      1.0000  1.0000  0.4741
   1(V4)  0.0000  0.0000  0.3720
   2      1.0000  1.0000  0.6280

       V1      V2      V3      V4
MRJ_98 EMRJ_98 CMRJ_98 OMRJ_98 SMRJ_98
MRJ_03 EMRJ_03 CMRJ_03 OMRJ_03 SMRJ_03
MRJ_08 EMRJ_08 CMRJ_08 OMRJ_08 SMRJ_08
```

Those output illustrates the likelihood of each response for classes within `SMK` (`RHO (a)`), `DRK` (`RHO (b)`), and `MRJ` (`RHO (c)`). High probabilities linked to specific responses guide the nature of each class. Therefore, we can label the classes for each variable as "Non-use" for class 1, "Occasional Use" for class 2, and "Heavy Use" for class 3. This labeling is informed by the response probabilities that indicate the extent of substance use associated with each class.

The focus then shifts to the `TAU` section, particularly `(D)`, `(E)`, and `(F)`. Following is corresponding part of the output:

```
TAU :

[...]

(D)
     parent
child       1       2       3       4       5
    1  0.9505  0.7958  0.1613  0.0254  0.0457
    2  0.0444  0.2042  0.6755  0.2230  0.2853
    3  0.0051  0.0000  0.1631  0.7516  0.6690

parent SUB_98 SUB_03 SUB_08
child  SMK_98 SMK_03 SMK_08
(E)
     parent
child       1       2       3       4       5
    1  0.7075  0.0065  0.1001  0.2107  0.0203
```

```
    2  0.2589  0.2414  0.6127  0.2229  0.1264
    3  0.0335  0.7521  0.2872  0.5664  0.8533

parent SUB_98 SUB_03 SUB_08
child  DRK_98 DRK_03 DRK_08
(F)
     parent
child       1       2       3       4       5
    1  0.9588  0.6173  0.1463  0.8512  0.0170
    2  0.0317  0.1525  0.7278  0.0871  0.1442
    3  0.0095  0.2302  0.1258  0.0617  0.8387

parent SUB_98 SUB_03 SUB_08
child  MRJ_98 MRJ_03 MRJ_08
```

These sections reveal the probability of belonging to `SMK`, `DRK`, `MRJ` classes given each `SUB_` class, respectively. For instance, because class 2 in `SUB_` is predominantly associated with high probabilities of "Heavy Use" in `DRK` but "Non-use" in `SMK` and `MRJ`, we can label it as "Heavy Drinking Only." We can label other `SUB` classes in the similar fashion: "Non-use" for class 1, "Occasional Use" for class 3, "Heavy Smoking & Drinking" for class 4, and "Heavy Use" for class 5, each reflective of the combined substance use behaviors.

Finally, the analysis moves to `PROF`, focusing on `PI` and `TAU` sections `(A)`, `(B)`, and `(C)`. Here is the corresponding part of the output:

```
PI :
(PROF)
  class
        1       2       3       4
   0.2195  0.2790  0.2869  0.2146

TAU :
(A)
     parent
child       1       2       3       4
    1  0.7018  0.3771  0.1825  0.1675
    2  0.0813  0.2620  0.0000  0.0000
    3  0.1766  0.2709  0.4491  0.3869
    4  0.0000  0.0000  0.1745  0.0000
    5  0.0403  0.0901  0.1939  0.4456

parent PROF
child  SUB_98
(B)
     parent
child       1       2       3       4
    1  0.9210  0.0000  0.0965  0.0000
    2  0.0711  0.9941  0.0000  0.0000
```

```
    3  0.0000  0.0059  0.0316  0.0269
    4  0.0000  0.0000  0.8719  0.0114
    5  0.0079  0.0000  0.0000  0.9617

parent PROF
child  SUB_03
(C)
     parent
child       1       2       3       4
    1  0.7168  0.0573  0.1844  0.0442
    2  0.2473  0.9311  0.0084  0.0356
    3  0.0040  0.0000  0.0048  0.0179
    4  0.0319  0.0019  0.8024  0.1821
    5  0.0000  0.0097  0.0000  0.7202

parent PROF
child  SUB_08
```

`PI` provides the prevalence of each profile class within `PROF`, while `TAU (A)`, `(B)`, and `(C)` show how these profiles are related to the substance use behaviors at different time points (i.e., `SUB_98`, `SUB_03`, and `SUB_08`). This analysis allows us to understand how substance use behaviors evolve over time and to label the `PROF` classes accordingly. For example, class 1 in `PROF` (i.e, parent class 1) might be labeled as "Non-use Stayers" because it consistently shows high probabilities for the "Non-use" class in `SUB` (i.e, child class 1) across all time points (i.e., in `TAU (A)`, `(B)`, and `(C)`. Similarly, other classes in `PROF` can be labeled as "Heavy Drinking Advancers" for class 2, "Heavy Smoking & Drinking Advancers" for class 3, and "Heavy-use Advancers" for class 4, reflecting the progressive intensity and diversity of substance use over time. Upon careful analysis of specific parts of the parameter estimates, we have effectively labeled each latent class, accurately reflecting the observed behaviors and their progression over time. This process enhances our understanding of substance use patterns and their longitudinal shifts.

6. Summary

This paper introduces the **slca** package in R, designed for advanced structural modeling with multiple latent class variables. Extending traditional latent class analysis (LCA), **slca** enables users to model complex relationships and dynamic changes over time. The package supports methodologies like latent transition analysis, latent class profile analysis, and joint latent class analysis, simplifying the construction of sophisticated models through a user-friendly interface. Empirical examples demonstrate the package's utility across different fields, such as sociopolitical attitudes, adolescent depression, and substance use patterns. By providing a unified framework for these methodologies, **slca** offers researchers a powerful tool for uncovering latent structures in data, making advanced statistical techniques more accessible.

In future research, several enhancements to the **slca** package could broaden its applicability in areas where LCA is frequently used. One potential improvement involves incorporating group effects into the models, which would be highly beneficial for researchers utilizing LCA. This

could be accomplished by integrating **slca** with the previously developed **glca** package (Kim *et al.* 2022), enabling the analysis of latent class differences across different groups. Another key direction is to address the boundary solution problem without resorting to parameter fixing, by adopting a Bayesian approach. This would offer more flexible and reliable estimation, particularly in the context of small sample sizes. Furthermore, extending the model to allow for multiple parent variables could accommodate more intricate dependencies among latent variables, thereby enriching the structural modeling framework.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grants RS-2024-00335008 and 2022M3J6A1063595 and the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant RS-2026-2550754 funded by the Korean government (MSIT) and Korea University Research Program K2511641 granted to Chung, and NRF grant RS-2026-25566081 granted to Kim.

References

Bache SM, Wickham H (2026). ***magrittr***: *A Forward-Pipe Operator for R*. doi:10.32614/CRAN.package.magrittr. R package version 2.0.5.

Bakk Z, Kuha J (2017). "Two-Step Estimation of Models between Latent Classes and External Variables." *Psychometrika*, **83**(4), 871–892. doi:10.1007/s11336-017-9592-7.

Bakk Z, Kuha J (2020). "Relating Latent Class Membership to External Variables: An Overview." *British Journal of Mathematical and Statistical Psychology*, **74**(2), 340–362. doi:10.1111/bmsp.12227.

Bartolucci F, Pandolfi S, Pennoni F (2017). "**LMest**: An R Package for Latent Markov Models for Longitudinal Categorical Data." *Journal of Statistical Software*, **81**(4), 1–38. doi:10.18637/jss.v081.i04.

Berndt ER, Hall BH, Hall RE, Hausman JA (1974). "Estimation and Inference in Nonlinear Structural Models." *Annals of Economic and Social Measurement*, **3**(4), 653–665.

Boker SM, Neale MC, Maes HH, Spiegel M, Brick TR, Estabrook R, Bates TC, Gore RJ, Hunter MD, Pritikin JN, Zahery M, Kirkpatrick RM (2026). ***OpenMx***: *Extended Structural Equation Modelling*. doi:10.32614/CRAN.package.OpenMx. R package version 2.22.11.

Bolck A, Croon M, Hagenaars J (2004). "Estimating Latent Structure Models with Categorical Variables: One-Step Versus Three-Step Estimators." *Political Analysis*, **12**(1), 3–27. doi:10.1093/pan/mph001.

Bureau of Labor Statistics, US Department of Labor (2026). "National Longitudinal Survey of Youth 1997 Cohort, 1997–2023 (Rounds 1–21)." *Technical report*, Center for Human Resource Research (CHRR), The Ohio State University, Columbus.

Chung H, Anthony JC, Schafer JL (2010). "Latent Class Profile Analysis: An Application to Stage Sequential Processes in Early Onset Drinking Behaviours." *Journal of the Royal Statistical Society A*, **174**(3), 689–712. doi:10.1111/j.1467-985x.2010.00674.x.

Collins LM, Lanza ST (2009). *Latent Class and Latent Transition Analysis: With Applications in the Social, Behavioral, and Health Sciences*. Wiley Series in Probability and Statistics. John Wiley & Sons. doi:10.1002/9780470567333.

Crouse MS, Nowak RD, Baraniuk RG (1998). "Wavelet-Based Statistical Signal Processing Using Hidden Markov Models." *IEEE Transactions on Signal Processing*, **46**(4), 886–902. doi:10.1109/78.668544.

Dempster AP, Laird NM, Rubin DB (1977). "Maximum Likelihood from Incomplete Data via the EM Algorithm." *Journal of the Royal Statistical Society B*, **39**(1), 1–22. doi:10.1111/j.2517-6161.1977.tb01600.x.

Dennis JE, Schnabel RB (1996). *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. SIAM. doi:10.1137/1.9781611971200.

Eddelbuettel D, François R (2011). "**Rcpp**: Seamless R and C++ Integration." *Journal of Statistical Software*, **40**(8), 1–18. doi:10.18637/jss.v040.i08.

Fop M, Murphy TB (2018). ***LCAvarsel**: Variable Selection for Latent Class Analysis*. doi:10.32614/CRAN.package.LCAvarsel. R package version 1.1.

Goodman LA (1974). "Exploratory Latent Structure Analysis Using Both Identifiable and Unidentifiable Models." *Biometrika*, **61**(2), 215–231. doi:10.1093/biomet/61.2.215.

Harris KM, Udry JR (2018). "National Longitudinal Study of Adolescent to Adult Health (Add Health), 1994–2008 [Public Use]." doi:10.3886/icpsr21600.v21.

Jeon S, Lee J, Anthony JC, Chung H (2017). "Latent Class Analysis for Multiple Discrete Latent Variables: A Study on the Association between Violent Behavior and Drug-Using Behaviors." *Structural Equation Modeling: A Multidisciplinary Journal*, **24**(6), 911–925. doi:10.1080/10705511.2017.1340844.

Jeon S, Seo TS, Anthony JC, Chung H (2020). "Latent Class Analysis for Repeatedly Measured Multiple Latent Class Variables." *Multivariate Behavioral Research*, **57**(2–3), 341–355. doi:10.1080/00273171.2020.1848515.

Kim EA, Chung H, Jeon S (2020). "Joint Latent Class Analysis for Longitudinal Data: An Application on Adolescent Emotional Well-Being." *Communications for Statistical Applications and Methods*, **27**(2), 241–254. doi:10.29220/csam.2020.27.2.241.

Kim Y, Chung H (2026). ***slca**: Structural Modeling for Multiple Latent Class Variables*. doi:10.32614/CRAN.package.slca. R package version 1.4.3.

Kim Y, Jeon S, Chang C, Chung H (2022). "**glca**: An R Package for Multiple-Group Latent Class Analysis." *Applied Psychological Measurement*, **46**(5), 439–441. ISSN 1552-3497. doi:10.1177/01466216221084197.

Lanza ST, Bray BC, Collins LM (2012). "An Introduction to Latent Class and Latent Transition Analysis." In *Handbook of Psychology*, 2nd edition. John Wiley & Sons. doi:10.1002/9781118133880.hop202024.

Lazarsfeld PF, Henry NW (1968). *Latent Structure Analysis*. Houghton, Mifflin.

Lee JW, Chung H (2017). "Latent Class Analysis with Multiple Latent Group Variables." *Communications for Statistical Applications and Methods*, **24**(2), 173–191. doi:10.5351/csam.2017.24.2.173.

Linzer DA, Lewis JB (2011). "**poLCA**: An R Package for Polytomous Variable Latent Class Analysis." *Journal of Statistical Software*, **42**(10), 1–29. doi:10.18637/jss.v042.i10.

Lo YT, Mendell NR, Rubin DB (2001). "Testing the Number of Components in a Normal Mixture." *Biometrika*, **88**(3), 767–778. doi:10.1093/biomet/88.3.767.

Lyrvall J, Mari RD, Bakk Z, Oser J, Kuha J (2025). "Multilevel Latent Class Analysis: State-of-the-Art Methodologies and Their Implementation in the R Package **multilevLCA**." *Multivariate Behavioral Research*, **60**(4), 731–747. doi:10.1080/00273171.2025.2473935.

McCutcheon A (1987). *Latent Class Analysis*. Sage Publications. doi:10.4135/9781412984713.

McCutcheon AL (1985). "A Latent Class Analysis of Tolerance for Nonconformity in the American Public." *Public Opinion Quarterly*, **49**(4), 474–488. doi:10.1086/268945.

McHugh RB (1956). "Efficient Estimation and Local Identification in Latent Class Analysis." *Psychometrika*, **21**(4), 331–347. doi:10.1007/bf02296300.

Muthén LK, Muthén BO (1998). *Mplus User's Guide*. URL https://www.statmodel.com/.

National Opinion Research Center (1992a). "General Social Survey, 1976." doi:10.3886/ICPSR07398.v1.

National Opinion Research Center (1992b). "General Social Survey, 1977." doi:10.3886/ICPSR07573.v1.

Nylund KL, Asparouhov T, Muthén BO (2007). "Deciding on the Number of Classes in Latent Class Analysis and Growth Mixture Modeling: A Monte Carlo Simulation Study." *Structural Equation Modeling: A Multidisciplinary Journal*, **14**(4), 535–569. doi:10.1080/10705510701575396.

R Core Team (2026). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. doi:10.32614/R.manuals. URL https://www.R-project.org/.

Rosseel Y (2012). "**lavaan**: An R Package for Structural Equation Modeling." *Journal of Statistical Software*, **48**(2), 1–36. doi:10.18637/jss.v048.i02.

Schnabel RB, Koonatz JE, Weiss BE (1985). "A Modular System of Algorithms for Unconstrained Minimization." *ACM Transactions on Mathematical Software*, **11**(4), 419–440. ISSN 1557-7295. doi:10.1145/6187.6192.

Vermunt JK (2010). "Latent Class Modeling with Covariates: Two Improved Three-Step Approaches." *Political Analysis*, **18**(4), 450–469. doi:10.1093/pan/mpq025.

Vermunt JK, Magidson J (2005). ***Latent GOLD*** *4.0 User's Guide*. Statistical Innovations.

White A, Murphy TB (2014). "**BayesLCA**: An R Package for Bayesian Latent Class Analysis." *Journal of Statistical Software*, **61**(13), 1–28. doi:10.18637/jss.v061.i13.

A. Derivation for updating formula

We begin by examining the Q-function in the expectation-maximization (EM) algorithm, which represents the expected complete-data log-likelihood given the observed data and the current parameter estimates. The Q-function is defined as:

$$\begin{aligned} Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(t)}) &= E_{\mathbf{X}|\mathbf{Y},\boldsymbol{\theta}^{(t)}} \left[\log L(\pi, \tau, \rho; \mathbf{X}, \mathbf{Y})\right] \\ &= \sum_{i=1}^{n} \sum_{\mathbf{x}} \mathsf{P}(\mathbf{X} = \mathbf{x} \mid \mathbf{Y} = \mathbf{y}_i, \boldsymbol{\theta}^{(t)}) \log L(\pi, \tau, \rho; \mathbf{y}_i) \\ &= \sum_{i=1}^{n} \sum_{\mathbf{x}} \mathsf{P}(\mathbf{X} = \mathbf{x} \mid \mathbf{Y} = \mathbf{y}_i, \boldsymbol{\theta}^{(t)}) \left[\sum_{r \in R} \log \pi_{x_r}^{(r)} + \sum_{j \in L} \log \tau_{x_j|\mathrm{pa}(x_j)}^{(j)} + \sum_{m \in M} \log \rho_{y_{im}|\mathrm{pa}(y_m)}^{(m)}\right] \end{aligned}$$

When optimizing with respect to a specific parameter, we focus on the term involving that parameter and sum out irrelevant latent variables. This simplification allows us to derive the update equations for each parameter. Using Lagrange multipliers to enforce the constraints that the probabilities sum to one, we can express the objective functions for each parameter. For the π parameters, the objective function is

$$Q(\pi_k^{(r)} \mid \boldsymbol{\theta}^{(t)}) = \sum_{i=1}^{n} \mathsf{P}(X_r = k \mid \mathbf{y}_i, \boldsymbol{\theta}^{(t)}) \log \pi_k^{(r)} - \lambda \left(\sum_k \pi_k^{(r)} - 1\right).$$

To optimize this function with respect to $\pi_k^{(r)}$, we take the derivative and set it to zero:

$$\frac{\partial Q(\pi_k^{(r)} \mid \boldsymbol{\theta}^{(t)})}{\partial \pi_k^{(r)}} = \sum_{i=1}^{n} \frac{\mathsf{P}(X_r = k \mid \mathbf{y}_i, \boldsymbol{\theta}^{(t)})}{\pi_k^{(r)}} - \lambda = 0$$

Solving for $\pi_k^{(r)}$, we have $\pi_k^{(r)} = \sum_{i=1}^{n} \mathsf{P}(X_r = k \mid \mathbf{y}_i, \boldsymbol{\theta}^{(t)})/\lambda$. To find the Lagrange multiplier λ, we use the constraint that the probabilities sum to one (i.e., $\sum_k \pi_k^{(r)} = 1$). Since for each i, the sum over k of $\mathsf{P}(X_r = k \mid \mathbf{y}_i, \boldsymbol{\theta}^{(t)})$ equals 1, summing over i from 1 to n gives $\lambda = n$. Therefore, the updated estimate for $\pi_k^{(r)}$ is

$$\pi_k^{(r)} = \frac{1}{n} \sum_{i=1}^{n} \mathsf{P}(X_r = k \mid \mathbf{y}_i, \boldsymbol{\theta}^{(t)}).$$

This shows that each $\pi_k^{(r)}$ is updated by averaging the posterior probabilities over all observations.

For the τ parameters, the objective function is

$$Q(\tau_{k|l}^{(j)} \mid \boldsymbol{\theta}^{(t)}) = \sum_{i=1}^{n} \mathsf{P}(X_j = k, \mathrm{pa}(X_j) = l \mid \mathbf{y}_i, \boldsymbol{\theta}^{(t)}) \log \tau_{k|l}^{(j)} - \lambda \left(\sum_k \tau_{k|l}^{(j)} - 1\right).$$

Differentiating with respect to $\tau_{k|l}^{(j)}$ and setting the derivative to zero, we obtain

$$\frac{\partial Q(\tau_{k|l}^{(j)} \mid \boldsymbol{\theta}^{(t)})}{\partial \tau_{k|l}^{(j)}} = \sum_{i=1}^{n} \frac{\mathsf{P}(X_l = k, \mathrm{pa}(X_j) = l \mid \mathbf{y}_i, \boldsymbol{\theta}^{(t)})}{\tau_{k|l}^{(j)}} - \lambda = 0.$$

Solving for $\tau_{k|l}^{(j)}$, we have $\tau_{k|l}^{(j)} = \sum_{i=1}^{n} \mathsf{P}(X_l = k, \text{pa}(X_j) = l \mid \mathbf{y}_i, \boldsymbol{\theta}^{(t)})/\lambda$. Using the constraint that the conditional probabilities sum to one for each parent state (i.e., $\sum_k \tau_{k|l}^{(j)} = 1$), we obtain $\lambda = \sum_{i=1}^{n} \mathsf{P}(\text{pa}(X_j) = l \mid \mathbf{y}_i, \boldsymbol{\theta}^{(t)})$. Note that X_l is marginalized by summing over k. Therefore, the updated estimate for $\tau_{k|l}^{(j)}$ is

$$\tau_{k|l}^{(j)} = \frac{\sum_{i=1}^{n} \mathsf{P}(X_l = k, \text{pa}(X_j) = l \mid \mathbf{y}_i, \boldsymbol{\theta}^{(t)})}{\sum_{i=1}^{n} \mathsf{P}(\text{pa}(X_j) = l \mid \mathbf{y}_i, \boldsymbol{\theta}^{(t)})}.$$

This formula updates $\tau_{k|l}^{(j)}$ by normalizing the expected counts of the joint states over the expected counts of the parent state.

For the ρ parameters, the objective function is

$$Q(\rho_{r|k}^{(m)} \mid \boldsymbol{\theta}^{(t)}) = \sum_{i=1}^{n} \mathsf{P}(\text{pa}(Y_m) = k \mid \mathbf{y}_i, \boldsymbol{\theta}^{(t)}) I(y_{im} = r) \log \rho_{r|k}^{(m)} - \lambda \left(\sum_r \rho_{r|k}^{(m)} - 1 \right),$$

where the indicator function $I(y_{im} = r)$ is 1 if $y_{im} = r$ or 0, otherwise. Taking the derivative with respect to $\rho_{r|k}^{(m)}$ and setting it to zero, we have

$$\frac{\partial Q(\rho_{r|k}^{(m)} \mid \boldsymbol{\theta}^{(t)})}{\partial \rho_{r|k}^{(m)}} = \sum_{i=1}^{n} \frac{\mathsf{P}(\text{pa}(Y_m) = k \mid \mathbf{y}_i, \boldsymbol{\theta}^{(t)}) I(y_{im} = r)}{\rho_{r|k}^{(m)}} - \lambda = 0.$$

Solving for $\rho_{r|k}^{(m)}$, we obtain $\rho_{r|k}^{(m)} = \sum_{i=1}^{n} \mathsf{P}(\text{pa}(Y_m) = k \mid \mathbf{y}_i, \boldsymbol{\theta}^{(t)}) I(y_{im} = r)/\lambda$. Using the constraint $\sum_r \rho_{r|k}^{(m)} = 1$, we can find $\lambda = \sum_{i=1}^{n} \mathsf{P}(\text{pa}(Y_m) = k \mid \mathbf{y}_i, \boldsymbol{\theta}^{(t)})$. Therefore, the updated estimate for $\rho_{r|k}^{(m)}$ is

$$\rho_{r|k}^{(m)} = \frac{\sum_{i=1}^{n} \mathsf{P}(\text{pa}(Y_m) = k \mid \mathbf{y}_i, \boldsymbol{\theta}^{(t)}) I(y_{im} = r)}{\sum_{i=1}^{n} \mathsf{P}(\text{pa}(Y_m) = k \mid \mathbf{y}_i, \boldsymbol{\theta}^{(t)})}.$$

Note that this formula updates $\rho_{r|k}^{(m)}$ by normalizing the weighted counts of observations where y_{im} equals r over the total expected counts of the parent state.

B. Upward-downward algorithm

The single-parent assumption described in Section 2 implies that the model's latent structure is a tree (or a forest of disconnected trees). The Upward-Downward algorithm leverages this property to efficiently compute the marginal posterior probabilities required in the E-step. To describe the subsets of observed variables used in the algorithm, we first define a general notation. Let $\mathcal{Y}_X$ be the set of observed variables that are descendants of an arbitrary latent node X (i.e., those depend on X itself or by its descendant latent variables). Using this definition, the total set of observed variables for the tree containing a node X_j is naturally denoted as $\mathcal{Y}_{\text{rt}(X_j)}$, where $\text{rt}(X_j)$ is the root of that tree. The posterior probability for $X_j$is thus conditioned on this complete set of relevant observed data:

$$\mathsf{P}(X_j = k \mid \mathbf{Y}) = \mathsf{P}(X_j = k \mid \mathcal{Y}_{\text{rt}(X_j)}) = \frac{\mathsf{P}(X_j = k, \mathcal{Y}_{\text{rt}(X_j)})}{\mathsf{P}(\mathcal{Y}_{\text{rt}(X_j)})}$$

The algorithm's strategy is to compute the joint probability by partitioning the set of observed variables around X_j. This partition consists of two subsets: the set of observed variables that are descendants of X_j, which is denoted $\mathcal{Y}_{X_j}$, and all other observed variables in the tree, denoted $\mathcal{Y}_{\mathrm{rt}(X_j)\setminus X_j}$. This allows the factorization of the joint probability into two components:

$$\mathsf{P}(X_j = k, \mathcal{Y}_{\mathrm{rt}(X_j)}) = \mathsf{P}(\mathcal{Y}_{X_j} \mid X_j = k) \cdot \mathsf{P}(X_j = k, \mathcal{Y}_{\mathrm{rt}(X_j)\setminus X_j})$$

The first term, $\mathsf{P}(\mathcal{Y}_{X_j} \mid X_j = k)$, is called the upward probability, denoted $\lambda_{X_j}(k)$. It is calculated recursively, beginning from the leaves of the tree and proceeding to the root. For notational simplicity in the recursive formulas, we define index sets for the children of a node X_j: $\mathcal{I}_X(X_j) = \{i \mid \mathrm{pa}(X_i) = X_j\}$ for latent children, and $\mathcal{I}_Y(X_j) = \{m \mid \mathrm{pa}(Y_m) = X_j\}$ for observed children. The process starts with the leaf nodes of the latent structure; for such a node X_j, its λ probability is the product of the measurement probabilities of its directly observed children:

$$\lambda_{X_j}(k) = \prod_{m \in \mathcal{I}_Y(X_j)} \rho^{(m)}_{r_m|k}$$

For a parent node, $\mathrm{pa}(X_j)$, the upward probability is computed recursively by aggregating the λ probabilities from its children nodes as follows:

$$\begin{aligned}
\lambda_{\mathrm{pa}(X_j)}(l) &= \mathsf{P}\left(\mathcal{Y}_{\mathrm{pa}(X_j)} \mid \mathrm{pa}(X_j) = l\right) \\
&= \left[\prod_{i \in \mathcal{I}_X(\mathrm{pa}(X_j))} \mathsf{P}\left(\mathcal{Y}_{X_i} \mid \mathrm{pa}(X_i) = l\right)\right] \prod_{m \in \mathcal{I}_Y(\mathrm{pa}(X_j))} \mathsf{P}\left(Y_m = r_m \mid \mathrm{pa}(X_j) = l\right) \\
&= \left[\prod_{i \in \mathcal{I}_X(\mathrm{pa}(X_j))} \sum_k \mathsf{P}\left(\mathcal{Y}_{X_i}, X_i = k \mid \mathrm{pa}(X_i) = l\right)\right] \prod_{m \in \mathcal{I}_Y(\mathrm{pa}(X_j))} \rho^{(m)}_{r_m|k} \\
&= \left[\prod_{i \in \mathcal{I}_X(\mathrm{pa}(X_j))} \sum_k \mathsf{P}\left(X_i = k \mid \mathrm{pa}(X_i) = l\right) P\left(\mathcal{Y}_{X_i} \mid X_i = k\right)\right] \prod_{m \in \mathcal{I}_Y(\mathrm{pa}(X_j))} \rho^{(m)}_{r_m|k} \\
&= \left[\prod_{i \in \mathcal{I}_X(\mathrm{pa}(X_j))} \sum_k \tau^{(i)}_{k|l} \lambda_{X_i}(k)\right] \prod_{m \in \mathcal{I}_Y(\mathrm{pa}(X_j))} \rho^{(m)}_{r_m|k},
\end{aligned}$$

Similarly, the second term, $\mathsf{P}(X_j = k, \mathcal{Y}_{\mathrm{rt}(X_j)\setminus X_j})$, is called the downward probability, denoted $\alpha_{X_j}(k)$, and is calculated recursively, starting from the root and proceeding down to the leaves. The recursion begins at the root node, X_0, where the α probability is defined as the root prevalence:

$$\alpha_{X_0}(k) = \mathsf{P}(X_0 = k) = \pi^{(0)}_k$$

For any non-root node X_j, the downward probability is then computed from its parent's

$(\mathrm{pa}(X_j))$ downward probability and the upward probabilities of its sibling nodes:

$$\begin{aligned}
\alpha_{X_j}(k) &= \mathsf{P}\left(\mathcal{Y}_{\mathrm{rt}(X_j)\setminus X_j}, X_j = k\right) \\
&= \sum_l \mathsf{P}\left(\mathcal{Y}_{\mathrm{rt}(X_j)\setminus X_j}, X_j = k, \mathrm{pa}(X_j) = l\right) \\
&= \sum_l \mathsf{P}\left(X_j = k \mid \mathrm{pa}(X_j) = l\right) \mathsf{P}\left(\mathcal{Y}_{\mathrm{rt}(X_j)\setminus X_j}, \mathrm{pa}(X_j) = l\right) \\
&= \sum_l \tau^{(j)}_{k|l} \mathsf{P}\left(\mathcal{Y}_{\mathrm{rt}(j)\setminus \mathrm{pa}(X_j)}, \mathrm{pa}(X_j) = l\right) \mathsf{P}\left(\mathcal{Y}_{\mathrm{pa}(X_j)\setminus X_j} \mid \mathrm{pa}(X_j) = l\right) \\
&= \sum_l \tau^{(j)}_{k|l} \alpha_{\mathrm{pa}(X_j)}(l) \frac{\mathsf{P}\left(\mathcal{Y}_{(\mathrm{pa}(X_j))} \mid \mathrm{pa}(X_j) = l\right)}{\mathsf{P}\left(\mathcal{Y}_{X_j} \mid \mathrm{pa}(X_j) = l\right)} \\
&= \sum_l \tau^{(j)}_{k|l} \alpha_{\mathrm{pa}(X_j)}(l) \frac{\lambda_{\mathrm{pa}(X_j)}(l)}{\sum_k \tau^{(j)}_{k|l} \lambda_{X_j}(l)}
\end{aligned}$$

Once both probabilities are computed for a node X_j, its marginal posterior is found by combining them and normalizing:

$$\mathsf{P}(X_j = k \mid \mathcal{Y}_{\mathrm{rt}(X_j)}) = \frac{\lambda_{X_j}(k)\alpha_{X_j}(k)}{\sum_{k'} \lambda_{X_j}(k')\alpha_{X_j}(k')}$$

The joint posterior of a variable and its parent, necessary for updating $\boldsymbol{\tau}$, is similarly computed:

$$\begin{aligned}
\mathsf{P}\left(X_j = k, \mathrm{pa}(X_j) = l \mid \mathcal{Y}_{\mathrm{rt}(X_j)}\right) &= \frac{\mathsf{P}\left(X_j = k, \mathrm{pa}(X_j) = l, \mathcal{Y}_{\mathrm{rt}(X_j)}\right)}{P\left(\mathcal{Y}_{\mathrm{rt}(X_j)}\right)} \\
&= \frac{\lambda_{X_j}(k)\tau^{(j)}_{k|l}\alpha_{\mathrm{pa}(X_j)}(l)\lambda_{\mathrm{pa}(X_j)}(l) \Big/ \sum_k \tau^{(j)}_{k|l}\lambda_{X_j}(k)}{\sum_{k'} \lambda_{X_j}(k')\alpha_{X_j}(k')}
\end{aligned}$$

C. Score function and Jacobian

The score vector $\mathbf{s}_i(\boldsymbol{\eta})$ is composed of the partial derivatives of the log-likelihood with respect to each free parameter. For the parameter groups in **slca**, these derivatives take a *posterior-prior* form. For an individual i, the derivatives for model parameters are

$$\begin{aligned}
\frac{\partial \log L_i}{\partial \psi^{(r)}_k} &= \mathsf{P}(X_r = k \mid \mathbf{y}_i) - \pi^{(r)}_k, \\
\frac{\partial \log L_i}{\partial \zeta^{(j)}_{k|l}} &= \mathsf{P}(X_j = k, \mathrm{pa}(X_j) = l \mid \mathbf{y}_i) - \tau^{(j)}_{k|l}\mathsf{P}(\mathrm{pa}(X_j) = l \mid \mathbf{y}_i), \text{ and} \\
\frac{\partial \log L_i}{\partial \omega^{(m)}_{r|k}} &= \mathsf{P}(\mathrm{pa}(Y_m) = k \mid \mathbf{Y} = \mathbf{y}_i)\left(I(y_{im} = r) - \rho^{(m)}_{r|k}\right),
\end{aligned}$$

where $I(\cdot)$ is the indicator function.

The Jacobian matrix J for the back-transformation is a block-diagonal matrix, with each block corresponding to an original parameter group $\boldsymbol{\pi}$, $\boldsymbol{\tau}$, and $\boldsymbol{\rho}$. For any parameter set (e.g., $\boldsymbol{\pi}^{(r)} = \{\pi_1^{(r)}, \ldots, \pi_K^{(r)}\}$) that follows a multinomial distribution, the inverse transformation from the $K-1$ free parameters back to the K probabilities is the softmax function. The entries of the corresponding Jacobian block are the partial derivatives of the softmax function:

$$\frac{\partial \theta_j}{\partial \eta_k} = \theta_j(\delta_{jk} - \theta_k),$$

where θ_j represents the jth element of an original parameter set, η_k represents the kth element of the corresponding free parameter set, and δ_{jk} equals 1 for $j = k$ and 0, otherwise.

Affiliation:

Youngsun Kim
Institute of Health and Environment
Seoul National University
Seoul, Republic of Korea
E-mail: yskstat@gmail.com

Hwan Chung
Department of Statistics
Korea University
Seoul, 02841, Republic of Korea
E-mail: hwanch@korea.ac.kr
URL: https://kustatlab.korea.ac.kr/

Journal of Statistical Software https://www.jstatsoft.org/
published by the Foundation for Open Access Statistics https://www.foastat.org/
October 2026, Volume 117, Issue 8 *Submitted:* 2024-12-19
doi:10.18637/jss.v117.i08 *Accepted:* 2025-09-15