lslx: Semi-Confirmatory Structural Equation Modeling via Penalized Likelihood

Po-Hsien Huang
National Cheng Kung University

Abstract

Sparse estimation via penalized likelihood (PL) is now a popular approach to learn the associations among a large set of variables. This paper describes an R package called **lslx** that implements PL methods for semi-confirmatory structural equation modeling (SEM). In this semi-confirmatory approach, each model parameter can be specified as free/fixed for theory testing, or penalized for exploration. By incorporating either a L_1 or minimax concave penalty, the sparsity pattern of the parameter matrix can be efficiently explored. Package **lslx** minimizes the PL criterion through a quasi-Newton method. The algorithm conducts line search and checks the first-order condition in each iteration to ensure the optimality of the obtained solution. A numerical comparison between competing packages shows that **lslx** can reliably find PL estimates with the least time. The current package also supports other advanced functionalities, including a two-stage method with auxiliary variables for missing data handling and a reparameterized multi-group SEM to explore population heterogeneity.

Keywords: structural equation modeling, factor analysis, penalized likelihood, R.

1. Introduction

For the past two decades, statistical modeling with sparsity has become a popular approach to learn associations among a large number of variables. In a sparse model, only a relatively small number of parameters play an important role (Hastie, Tibshirani, and Wainwright 2015). From a substantive view point, it is easier to interpret a sparse model than a dense one. From a statistical view point, a sparse model can yield an estimator with smaller mean squared error (e.g., Knight and Fu 2000; Negahban, Ravikumar, Wainwright, and Yu 2012). Since the exact sparsity pattern of a model is generally unknown in advance, the model is often probed by a sparse estimation procedure with penalization (or regularization; e.g., Tibshirani 1996; Fan and Li 2001; Zhang 2010). Package **glmnet** (Friedman, Hastie, and Tibshirani 2010) and library **LIBLINEAR** (Fan, Chang, Hsieh, Wang, and Lin 2008) are well-known software solutions for sparse linear models with regularization.

Psychometric models mostly impose a strong sparsity assumption for identification or interpretation purpose (e.g., Thurstone 1947). Recently, several penalized likelihood (PL) based data-driven methods have been proposed to depict sparsity patterns in psychometric models. (e.g., Chen, Liu, Xu, and Ying 2015; Hirose and Yamamoto 2015, 2014; Huang, Chen, and Weng 2017; Jacobucci, Grimm, and McArdle 2016; Tutz and Schauberger 2015). The present work describes an R (R Core Team 2017) package called **lslx** (latent structure learning

extended) that implements PL methods for structural equation modeling (SEM), one of the most popular multivariate techniques for psychological studies (Hershberger 2003).

There are many popular software packages for conducting SEM analysis, including commercial programs like LISREL (Jöreskog and Sörbom 2015), EQS (Bentler 2006), and Mplus (Muthén and Muthén 1998-2010), as well as non-commercial R packages such as **sem** (Fox, Nie, Byrnes, Culbertson, DebRoy, Friendly, Goodrich, Jones, Kramer, Monette, and R-Core 2017), **lavaan** (Rosseel 2012), and **OpenMx** (Neale, Hunter, Pritikin, Zahery, Brick, Kirkpatrick, Estabrook, Bates, Maes, and Boker 2016). However, none of them can directly conduct sparse estimation via PL. It might be a challenging task to incorporate PL methods in these well-developed software solutions because PL requires (1) a modified syntax for model specification; (2) a re-designed algorithm for optimizing non-differentiable objective functions; and (3) a new data-structure to store fitting results.

Before **lslx**, there were four packages that could fit SEM-related models via PL: **sparseSEM** (Huang 2014), **lsl** (Huang 2017b), **regsem** (Jacobucci, Grimm, Brandmaier, and Serang 2017), and **lvnet** (Epskamp 2017). However, only **lsl** and **regsem** were able to fit the commonly used class of SEM models. Package **sparseSEM** cannot handle latent variables (Cai, Bazerque, and Giannakis 2013), while package **lvnet** mainly utilizes PL to explore the precision matrix of the latent factors or residuals (Epskamp, Rhemtulla, and Borsboom 2017).

Package **lsl** employs the PL method developed by Huang *et al.* (2017), and it is a predecessor of **lslx**. It supports SEM models that can be represented by the Jöreskog-Keesling-Wiley model (Keesling 1972; Jöreskog 1973; Wiley 1973) via matrix specifications. Except for variance parameters, every coefficient can be set as free, fixed, or penalized. The solution path of the PL estimates can be obtained by an expectation-conditional maximization (ECM) algorithm (Meng and Rubin 1993). However, **lsl** has two major drawbacks: (1) model specification through pattern matrices is not user-friendly; (2) the optimization via the ECM algorithm has only linear convergence (Meng 1994). In addition, since ECM relies on the functional form of normal theory likelihood, it cannot be extended to other types of fitting functions. Package **regsem** implements the regularized SEM proposed by Jacobucci *et al.* (2016), adopting the reticular action model (RAM; McArdle and McDonald 1984) as a framework for model representation. Because it receives the outputs from **lavaan** for subsequent PL analysis, model specification in **regsem** is relatively convenient. The central problem with **regsem** is that its optimization algorithm often misses optimal solutions. A detailed account of this issue is presented in Section 6.

Package **lslx** was constructed to overcome the drawbacks of existing packages for SEM with PL. The author believes that **lslx** has at least four advantages: (1) Model specification is relatively easy. It adopts a **lavaan**-like model syntax with carefully designed operators and prefixes. Through the model syntax, users can set each coefficient as free, fixed, or penalized. When the syntax is not convenient enough, built-in methods can be also used to modify the initially specified model. (2) The optimization algorithm in **lslx** is reliable. Motivated by the works of Friedman *et al.* (2010) and Yuan, Ho, and Lin (2012), **lslx** implemented a quasi-Newton method that conducts line-search and checks the first-order condition in every iteration to ensure optimality. Furthermore, related numerical conditions can be plotted to monitor the optimization quality. (3) **lslx** is reasonably fast. The implemented quasi-Newton algorithm can achieve locally superlinear convergence under suitable conditions (Yuan *et al.* 2012). In addition, the core of **lslx** is written via **Rcpp** (Eddelbuettel and François 2011) and **RcppEigen** (Bates and Eddelbuettel 2013). As we shall see in Section 6, **lslx** is significantly

faster than both **lsl** and **regsem**. (4) **lslx** has several additional functionalities. Like usual SEM packages, **lslx** provides formal statistical test results, including tests for goodness-of-fit and coefficients. Besides, **lslx** handles missing data via a two-stage method with auxiliary variables (Savalei and Bentler 2009), and conducts multi-group analysis with a reparameterized SEM formulation (Huang in press).

This paper is organized as follows. Section 2 describes the PL method for semi-confirmatory SEM. In Section 3, a quasi-Newton algorithm for optimizing the PL criterion is introduced. Section 4 demonstrates how to implement the semi-confirmatory SEM with **lslx**. In Section 5, advanced functionalities for **lslx** are described, including a two-stage method for missing data and multi-group SEM analysis. Section 6 presents a numerical comparison. Finally, merits, limitations, and further directions concerning **lslx** are discussed.

2. Semi-confirmatory SEM and PL

In this section, the SEM formulation adopted for **lslx** and the theory of a semi-confirmatory SEM via PL are described.

2.1. SEM and theory testing

Let η denote a $(P+M)$-dimensional random column vector, which we partition into a P-dimensional observable response y and an M-dimensional latent factor f, that is, $\eta^\top = (y^\top, f^\top)$. In **lslx**, the following SEM model formulation is adopted

$$\eta = \alpha + \mathrm{B}\eta + \zeta, \tag{1}$$

where α is a $(P+M)$-dimensional intercept, B is a $(P+M)\times(P+M)$ regression coefficient matrix, and ζ is a $(P+M)$-dimensional random residual with zero mean and covariance matrix Φ. Let θ denote a Q-dimensional parameter vector with general component θ_q. The parameter vector contains the non-trivial elements from α, B, and Φ. Under the assumption that $(I-\mathrm{B})^{-1}$ exists, the model-implied mean and covariance of y can be written as

$$\begin{aligned} \mu(\theta) &= G(I-\mathrm{B})^{-1}\alpha, \\ \Sigma(\theta) &= G(I-\mathrm{B})^{-1}\Phi{(I-\mathrm{B})^{-1}}^\top G^\top, \end{aligned} \tag{2}$$

where I is the identity matrix and G is a selection matrix such that $y = G\eta$. Equation 1 and 2 can be understood as the RAM formulation (McArdle and McDonald 1984) covering the well-known Jöreskog-Keesling-Wiley model (Keesling 1972; Jöreskog 1973; Wiley 1973) and the Bentler-Weeks model (Bentler and Weeks 1980). Many statistical techniques can be represented in this framework, including linear regression with random covariates, path analysis, factor analysis, and growth curve models. Note that **lslx** users are not required to understand how Equation 1 and 2 represent any of these models. They only need to correctly specify the relationship between y and f via operators and variables (see Section 4.1 for model syntax).

The aim of a SEM analysis is to verify if there exists a θ^* such that the population mean and the covariance of y are closely represented by the implied moments, i.e., $\mu \approx \mu(\theta^*)$ and $\Sigma \approx \Sigma(\theta^*)$. Because μ, Σ, and θ^* are unknown population quantities, their sample counterparts m, S, and $\hat{\theta}$ are considered for actual calculations. Given a random sample

$\mathcal{Y} = \{y_n\}_{n=1}^N$ of size N, the most commonly used estimation procedure is to minimize the maximum likelihood (ML) loss function

$$\mathcal{D}(\theta) = \text{tr}(S\Sigma(\theta)^{-1}) - \log|S\Sigma(\theta)^{-1}| - P + [m - \mu(\theta)]^\top \Sigma(\theta)^{-1}[m - \mu(\theta)], \tag{3}$$

where $m = \frac{1}{N}\sum_{n=1}^N y_n$ and $S = \frac{1}{N}\sum_{n=1}^N (y_n - m)(y_n - m)^\top$. The plausibility of the specified model can be evaluated by the likelihood ratio (LR) test on $N\mathcal{D}(\hat{\theta})$ and by many other goodness-of-fit indices (see West, Taylor, and Wu 2012, for a review). If the specified model doesn't fit data well, the model should be abandoned.

In practice, an initially specified model is rarely abandoned immediately. Jöreskog (1993) distinguished three situations for applying SEM: strict theory confirmation, model comparison, and model generation. He argued that model generation is the most common case. Under model generation, users successively improve the initially specified model via modification indices (e.g., Sörbom 1989) or other strategies. Discussing whether a confirmatory or exploratory study is more appropriate is beyond the scope of this paper. From the author's perspective, however, several instances of SEM analysis are both confirmatory and exploratory. On the one hand, the analyst aims to test the core hypotheses in the specified model, on the other hand, the analyst seeks an optimal pattern for the exploratory part to avoid the price of model misspecification (e.g., Bentler and Mooijaart 1989; Yuan, Marshall, and Bentler 2003). The author's preference is best called a semi-confirmatory approach.

2.2. PL for semi-confirmatory SEM

Huang *et al.* (2017) proposed a semi-confirmatory method for SEM via PL. In this method, a SEM model contains two parts: a confirmatory part and an exploratory part. The confirmatory part includes all freely estimated parameters and fixed parameters that are allowed for theory testing. The exploratory part is composed of a set of penalized parameters to describe relations that cannot be determined by existing substantive theory. By implementing a sparsity-inducing penalty and choosing an optimal penalty level, the relationships in the exploratory part can be efficiently determined. This semi-confirmatory method can be seen as a methodology that links the traditional SEM with the exploratory SEM (Asparouhov and Muthén 2009).

To conduct the semi-confirmatory approach for SEM, **lslx** considers the following optimization problem

$$\min_{\theta} \quad \mathcal{U}(\theta, \lambda), \tag{4}$$

where $\mathcal{U}(\theta, \lambda)$ is a PL objective function with the form

$$\mathcal{U}(\theta, \lambda) = \mathcal{D}(\theta) + \mathcal{R}(\theta, \lambda), \tag{5}$$

$\mathcal{R}(\theta, \lambda)$ is a penalty term (or regularizer), and $\lambda > 0$ is a regularization parameter to control the penalty level. In particular, the penalty term can be written as $\mathcal{R}(\theta, \lambda) = \sum_{q=1}^Q c_{\theta_q}\rho(|\theta_q|, \lambda)$ with $\rho(|\theta_q|, \lambda)$ being a penalty function and $c_{\theta_q} \in \{0, 1\}$ being an indicator to show whether θ_q is penalized. The current version of **lslx** implements the minimax concave penalty (MCP; Zhang 2010)

$$\rho(|\theta_q|, \lambda) = \begin{cases} \lambda|\theta_q| - \frac{\theta_q^2}{2\delta} & \text{if } |\theta_q| \leq \lambda\delta, \\ \frac{1}{2}\lambda^2\delta & \text{if } \lambda\delta < |\theta_q|, \end{cases} \tag{6}$$

where δ is a parameter to control the convexity of MCP. The MCP produces a sparse estimator, and has many good theoretical properties (see Mazumder, Friedman, and Hastie 2011; Zhang 2010). If $\delta \to \infty$, the MCP reduces to the case of L_1 penalty or LASSO (least absolute shrinkage and selection operator; Tibshirani 1996). On the other hand, a small value of δ makes the MCP behave like hard thresholding. In linear regression with standardized covariates, δ must be larger than one. However, when incorporating MCP in SEM, the lower bound of δ depends on the Hessian matrix of the ML loss function (see Section 3.2).

Given a penalty level λ and a convexity level δ, a PL estimator $\hat{\theta} \equiv \hat{\theta}(\lambda, \delta)$ is defined as a solution of Equation 4. In practice, an optimal pair of (λ, δ), denoted by $(\hat{\lambda}, \hat{\delta})$, is often selected by minimizing an information criterion (or cross-validation error) over $\Lambda \times \Delta$, where Λ and Δ are two user-defined sets formed by $\lambda_1 < \lambda_2 < ... < \lambda_K$ and $\delta_1 < \delta_2 < ... < \delta_L$, respectively. **lslx** utilizes several information criteria that can be written as

$$IC(\hat{\theta}) = \mathcal{D}(\hat{\theta}) - C_N df(\hat{\theta}), \tag{7}$$

where C_N is a sequence that depends on sample size N and $df(\hat{\theta})$ is the degrees of freedom. In a usual case, $df(\hat{\theta})$ is calculated by $P(P+3)/2 - e(\hat{\theta})$ with $e(\hat{\theta})$ being the number of non-zero elements in $\hat{\theta}$. The Bayesian information criterion (BIC; Schwarz 1978) corresponds to the case when $C_N = \log(N)/N$ and the Akaike information criterion (AIC; Akaike 1974) corresponds to the case when $C_N = 2/N$. Other information criteria include AIC with penalty 3 (AIC3; Sclove 1987), consistent AIC (CAIC; Bozdogan 1987), adjusted BIC (ABIC; Sclove 1987), and Haughton's BIC (HBIC; Haughton 1988). In addition, a robust version of information criterion is also available in **lslx**. The robust version is taken as the usual information criterion with degrees of freedom being replaced by the expectation of LR statistics under general conditions (e.g., Yuan, Hayashi, and Bentler 2007, see Appendix A for technical details).

After $(\hat{\lambda}, \hat{\delta})$ is determined, the appropriateness of the final model provided by $\hat{\theta} \equiv \hat{\theta}(\hat{\lambda}, \hat{\delta})$ should be evaluated. The model fit can be assessed by the usual fit indices calculated from $\hat{\theta}$. The significance of the non-zero elements of $\hat{\theta}$ can be also tested by sandwich standard errors (e.g., Browne 1984; Yuan and Hayashi 2006; Yuan and Lu 2008, see Appendix A). However, classical statistical inferences are generally incorrect after penalty level selection (or any model selection process; see Leeb and Pötscher 2006; Pötscher 1991). An exception is if the procedure can yield an oracle estimator (i.e., an estimator performs as well as if the true sparsity pattern is known in advance), the associated statistical inferences become valid. It has been shown that PL with MCP and BIC selector asymptotically results in an oracle estimator, both theoretically and empirically (Huang *et al.* 2017). Despite this, the oracle property might not hold under small sample size scenarios. Users should be cautious about the hypothesis testing and confidence interval results for the penalized parameters.

The overall procedure of semi-confirmatory SEM via PL is shown in Algorithm 1.

3. Optimization algorithm

In **lslx**, a quasi-Newton method is implemented to solve the problem in Equation 4. The method is established based on Yuan *et al.* (2012) who modified the coordinate descent algorithm in **glmnet** (Friedman *et al.* 2010) to ensure its convergence for L_1-penalized logistic regression. This section describes how this modified algorithm can be extended to minimize a PL criterion with MCP for SEM. Roughly speaking, the algorithm consists of outer itera-

Algorithm 1: Semi-confirmatory structural equation modeling via penalized likelihood.

Model specification: determine which elements in α, B, and Φ should be free, fixed, or penalized.
Data preparation: input a data set $\mathcal{Y}$.
Initialization: specify $\Lambda = \{\lambda_k\}_{k=1}^{K}$ with $\lambda_k < \lambda_{k+1}$ and $\Delta = \{\delta_l\}_{l=1}^{L}$ with $\delta_l < \delta_{l+1}$.
Estimation:
for $k = 1, 2, ..., K$ *(or* $k = K, K-1, ..., 1$*)* **do**
  for $l = L, L-1, ..., 1$ **do**
    compute a PL estimate $\hat{\theta} \equiv \hat{\theta}(\lambda_k, \delta_l)$ by solving Equation 4.
Selection and evaluation: choose $(\hat{\lambda}, \hat{\delta})$ by minimizing some $IC(\hat{\theta})$ and evaluate the model made by $\hat{\theta} \equiv \hat{\theta}(\hat{\lambda}, \hat{\delta})$.

tions and inner iterations. The implementation of the quasi-Newton method is summarized in Algorithm 2.

3.1. Outer iteration

Let $\hat{\theta}^{(t)} \equiv \hat{\theta}^{(t)}(\lambda, \delta)$ denote the estimate at the t^{th} outer iteration under λ and δ. Suppose a corresponding quasi-Newton direction $\hat{d}$ is available. The outer iteration aims to find a step size $\hat{s}$ such that the updated estimate $\hat{\theta}^{(t+1)} = \hat{\theta}^{(t)} + \hat{s} \times \hat{d}$ satisfies

$$\mathcal{U}(\hat{\theta}^{(t)} + \hat{s} \times \hat{d}, \lambda) - \mathcal{U}(\hat{\theta}^{(t)}, \lambda) \leq c_{\text{Armijo}} \times \hat{s} \times \left(\frac{\partial \mathcal{D}(\hat{\theta}^{(t)})}{\partial \theta^{\top}} \hat{d} + \mathcal{R}(\hat{\theta}^{(t)} + \hat{d}, \lambda) - \mathcal{R}(\hat{\theta}^{(t)}, \lambda) \right), \quad (8)$$

where $0 < c_{\text{Armijo}} < 1$ is a specified Armijo's constant. With some given $0 < s < 1$, **lslx** adopts $\hat{s}$ as the largest element in $\{s^j\}_{j=0}^{J}$ such that Equation 8 is satisfied.

According to the first-order optimality condition, a PL estimate $\hat{\theta}$ should satisfy

$$\frac{\partial \mathcal{U}(\hat{\theta}, \lambda)}{\partial \theta_q} \equiv \begin{cases} \frac{\partial \mathcal{D}(\hat{\theta})}{\partial \theta_q} - \frac{\partial \mathcal{R}(\hat{\theta}, \lambda)}{\partial \theta_q} = 0 & \text{if } \hat{\theta}_q \neq 0 \text{ or } c_{\theta_q} = 0, \\ \text{sign}(\frac{\partial \mathcal{D}(\hat{\theta})}{\partial \theta_q}) \max\left\{ |\frac{\partial \mathcal{D}(\hat{\theta})}{\partial \theta_q}| - \lambda, 0 \right\} = 0 & \text{if } \hat{\theta}_q = 0 \text{ and } c_{\theta_q} = 1. \end{cases} \quad (9)$$

where $\text{sign}(\cdot)$ is a function that extracts the sign of a number. Note that $\mathcal{U}(\theta, \lambda)$ is not differentiable in the usual sense if $\theta_q = 0$ for some $c_{\theta_q} = 1$. $\frac{\partial \mathcal{U}(\hat{\theta}, \lambda)}{\partial \theta_q}$ actually represents the q^{th} component of the sub-gradient of $\mathcal{U}(\theta, \lambda)$ evaluated at $\hat{\theta}$. In **lslx**, the outer iteration stops if the following condition is satisfied.

$$\max_q \left| \frac{\partial \mathcal{U}(\hat{\theta}, \lambda)}{\partial \theta_q} \right| \leq \epsilon_{\text{out}}, \quad (10)$$

where $\epsilon_{\text{out}} > 0$ is a specified tolerance for convergence of the outer loop. Let $\text{vech}(\cdot)$ denote an operator that stacks non-duplicated elements of a symmetric matrix. The sub-gradient can be obtained by

$$\frac{\partial \mathcal{D}(\theta)}{\partial \theta} = 2 \left(\frac{\partial \tau(\theta)}{\partial \theta^{\top}} \right)^{\top} W(\theta) \begin{pmatrix} m - \mu(\theta) \\ \text{vech}[S + (m - \mu(\theta))(m - \mu(\theta))^{\top} - \Sigma(\theta)] \end{pmatrix}, \quad (11)$$

and

$$\frac{\partial \mathcal{R}(\theta, \lambda)}{\partial \theta_q} = \begin{cases} \text{sign}(\theta_q)\lambda - \frac{\theta_q}{\delta} & \text{if } |\theta_q| \leq \lambda\delta, \\ 0 & \text{if } \lambda\delta < |\theta_q|, \end{cases} \tag{12}$$

where $\tau(\theta) = \begin{pmatrix} \mu(\theta) \\ \sigma(\theta) \end{pmatrix}$ and $W(\theta) = \begin{pmatrix} \Sigma(\theta)^{-1} \\ \frac{1}{2} D_P^\top [\Sigma(\theta)^{-1} \otimes \Sigma(\theta)^{-1}] D_P \end{pmatrix}$ with $\sigma(\theta) = \text{vech}(\Sigma(\theta))$ and D_P being the duplication matrix of size $PP \times P(P+1)/2$ (see Magnus and Neudecker 1999). The specific form of the model Jacobian $\frac{\partial \tau(\theta)}{\partial \theta^\top}$ can be found in Bentler and Weeks (1980) and in Neudecker and Satorra (1991).

The success of the outer iteration relies on a good starting value $\hat{\theta}^{(0)}$. In **lslx**, if $\hat{\theta}(\lambda_k, \delta_l)$ is computed after deriving a PL estimate in the neighborhood of (λ_k, δ_l), $\hat{\theta}^{(0)}(\lambda_k, \delta_l)$ will be set as $\hat{\theta}(\lambda_k, \delta_{l+1})$ or $\hat{\theta}(\lambda_{k-1}, \delta_l)$ for a warm start. The warm start approach makes $\hat{\theta}(\lambda_k, \delta_l)$ readily available (see also Friedman *et al.* 2010). If no appropriate PL estimate is available, a default $\hat{\theta}^{(0)}$ is calculated by the method of McDonald and Hartmann (1992). The author's experience shows that this method works well if the scales of the response variables are similar, and the variances of the latent factors are approximately one.

3.2. Inner iteration

Consider the following quadratic approximation for $\mathcal{U}(\hat{\theta}^{(t)} + d, \lambda) - \mathcal{U}(\hat{\theta}^{(t)}, \lambda)$

$$\mathcal{Q}^{(t)}(d) = g^{(t)\top} d + \frac{1}{2} d^\top H^{(t)} d + \mathcal{R}(\hat{\theta}^{(t)} + d, \lambda) - \mathcal{R}(\hat{\theta}^{(t)}, \lambda), \tag{13}$$

where $g^{(t)}$ and $H^{(t)}$ denote the gradient and the Hessian matrix (or an approximation thereof) of $\mathcal{D}(\hat{\theta}^{(t)})$, respectively. The inner iteration looks for a quasi-Newton direction $\hat{d}$ by minimizing $\mathcal{Q}^{(t)}(d)$. Because the quadratic approximation is not differentiable at θ when $\theta_q = 0$ for some $c_{\theta_q} = 1$, the proposed quasi-Newton method minimizes Equation 13 by a coordinate descent method. Let $\hat{d}^{(r+(q-1)/Q)}$ denote the estimated direction at inner iteration $r + (q-1)/Q$. The inner iteration updates $\hat{d}^{(r+(q-1)/Q)}$ by $\hat{d}^{(r+(q-1)/Q)} + \hat{z}_q \times e_q$ with an appropriate step size of $\hat{z}_q$, where e_q is the q^{th} vector in the standard basis.

The step size of $\hat{z}_q$ can be obtained by solving the following univariate problem

$$\min_{z_q} \quad f(z_q), \tag{14}$$

where

$$\begin{aligned} f(z_q) &= \mathcal{Q}^{(t)}(\hat{d}^{(r+(q-1)/Q)} + z_q \times e_q) - \mathcal{Q}^{(t)}(\hat{d}^{(r+(q-1)/Q)}) \\ &= a z_q + \frac{1}{2} b z_q^2 + \rho(|c + z_q|, \lambda) - \rho(|c|, \lambda), \end{aligned} \tag{15}$$

with $a = g_q^{(t)} + (H^{(t)} \hat{d}^{(r+q/Q)})_q$, $b = H_{qq}^{(t)}$, and $c = \hat{\theta}_q^{(t)} + \hat{d}_q^{(r+(q-1)/Q)}$. Under MCP penalty, the solution of Equation 14 is

$$\hat{z}_q = \begin{cases} \frac{-a}{b} & \text{if } \frac{c/\delta - a - \lambda}{b - 1/\delta} \geq \lambda\delta - c, \\ \frac{c/\delta - a - \lambda}{b - 1/\delta} & \text{if } \lambda\delta - c \geq \frac{c/\delta - a - \lambda}{b - 1/\delta} \geq -c, \\ -c & \text{otherwise}, \\ \frac{c/\delta - a + \lambda}{b - 1/\delta} & \text{if } -\lambda\delta - c \leq \frac{c/\delta - a + \lambda}{b - 1/\delta} \leq -c, \\ \frac{-a}{b} & \text{if } \frac{c/\delta - a + \lambda}{b - 1/\delta} \leq -\lambda\delta - c. \end{cases} \tag{16}$$

Algorithm 2: Quasi-Newton method for solving penalized likelihood with MCP.

Initialization: initialize $\hat{\theta}^{(0)} \equiv \hat{\theta}^{(0)}(\lambda, \delta)$ with the given λ and δ, and specify $\epsilon_{\text{in}} > 0$ and $\epsilon_{\text{out}} > 0$.
Outer iteration:
for $t = 0, 1, 2, ...$ **do**
  Inner iteration:
  set $\hat{d}^{(0)} = 0$.
  for $r = 0, 1, 2, ...$ **do**
    for $q = 1, 2, ..., Q$ **do**
      compute $\hat{z}_q$ by solving Equation 14 and set $\hat{d}^{(r+q/Q)} = \hat{d}^{(r+(q-1)/Q)} + \hat{z}_q \times e_q$.
    if $\max_q \left| H^{(t)}_{qq} \hat{z}_q \right| \leq \epsilon_{\text{in}}$ **then**
      output $\hat{d} = \hat{d}^{(r+1)}$.
      break
  find $\hat{s}$ satisfying Equation 8 and set $\hat{\theta}^{(t+1)} = \hat{\theta}^{(t)} + \hat{s} \times \hat{d}$.
  if $\max_q \left| \frac{\partial \mathcal{U}(\hat{\theta}^{(t+1)}, \lambda)}{\partial \theta_q} \right| \leq \epsilon_{\text{out}}$ **then**
    output $\hat{\theta} \equiv \hat{\theta}(\lambda, \delta) = \hat{\theta}^{(t+1)}$.
    break

If no penalty is imposed for θ_q, $\hat{z}_q$ is simply $\frac{-a}{b}$. It should be noted that the problem in Equation 15 is convex if and only if $b - \frac{1}{\delta} > 0$. For $b - \frac{1}{\delta} \leq 0$, the proposed algorithm may fail. As the value of b is generally unknown in advance, it is better to start with larger values of δ.

Let $\hat{z}_1$, $\hat{z}_2$, ..., $\hat{z}_Q$ denote the obtained directions for some inner iteration. In **lslx**, the inner looping stops if

$$\max_q \left| H^{(t)}_{qq} \hat{z}_q \right| \leq \epsilon_{\text{in}}, \tag{17}$$

where $\epsilon_{\text{in}} > 0$ is a specified tolerance for the inner iteration. The implementation of the inner iteration relies on the choice of $H^{(t)}$. The exact Hessian is generally too expensive to be calculated. A natural choice for replacement is the expected Hessian (or the expected Fisher information matrix)

$$H^{(t)} = 2 \left(\frac{\partial \tau(\hat{\theta}^{(t)})}{\partial \theta^\top} \right)^\top W(\hat{\theta}^{(t)}) \frac{\partial \tau(\hat{\theta}^{(t)})}{\partial \theta^\top}, \tag{18}$$

which results in a Fisher scoring-type algorithm. A simple alternative approximation is the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method, which yields a BFGS type algorithm. Based on the results of the numerical comparison in Section 6, the two choices for $H^{(t)}$ yield similar performances in terms of computation time.

4. lslx package

This section describes how to specify a model and conduct semi-confirmatory SEM under **lslx**. The main function `lslx` is a class generator to initialize an `lslx` object. Object `lslx` is constructed via R6 system (Chang 2017), adopting encapsulation object-oriented programming style. Despite that commonly used S3 or S4 system is also capable of constructing an

equivalent object, the author adopts R6 for mainly two reasons. First, R6 methods belong to objects, which means that it is straightforward to create many small functions to manipulate objects under R6. Second, the mutability of R6 allows us to develop methods to modify the initially generated object without returning a new one. It is particularly useful when a user hope to modify an initially specified model by freeing, fixing, or penalizing elements in some block of a coefficient matrix. As we shall see later, `lslx` objects can be flexibly manipulated through many build-in methods with a consistent naming scheme.

In the simplest case, the use of `lslx` object involves three major steps

1. Initialize a new `lslx` object by specifying a model and a data set.

   ```
   r6_lslx <- lslx$new(model, data)
   ```

2. Fit the specified model to the data with given fitting options.

   ```
   r6_lslx$fit(penalty_method, lambda_grid, delta_grid)
   ```

3. Summarize the fitting results using the specified selector.

   ```
   r6_lslx$summarize(selector)
   ```

By default, **lslx** implements MCP as penalty function (`penalty_method = "mcp"`) with Λ and Δ constructed by the method described in Section 4.4 and Appendix B. The other arguments are all required to be specified (i.e., no default values). After an `lslx` object is initialized, a print method can be used (e.g., `r6_lslx$print()`) to hint on how the object can be further manipulated.

4.1. Model syntax

The model syntax in **lslx** is highly motivated by **lavaan** (Rosseel 2012). There the relationships among observed responses and latent factors are specified via an equation-like syntax. To demonstrate the model syntax, consider the following example for a regression model with one dependent variable (`y`) and four covariates (`x1 - x4`)

```
R> model_reg <- "y <= x1 + x2
+                y <~ x3 + x4"
```

Here, the operator `<=` indicates that the regression coefficients from the right-hand side (RHS) variables should be set as freely estimated parameters. On the other hand, `<~` makes the coefficients from the RHS variables to be estimated with penalization. The distinction between `<=` and `<~` in **lslx** can help users set the types for each coefficient. The example model estimates coefficients from `x1` and `x2` to `y` freely, and it sets coefficients from `x3` and `x4` to `y` as penalized. In addition, the model implicitly sets the following coefficients to be free: (1) the variances of `x1 - x4` and the residual variance of `y`; (2) the covariances among `x1 - x4`; and (3) the intercepts of `x1 - x4` and `y`.

Through the previous example, we can see that in **lslx** a model can be formulated by simply classifying the (parameter) types for the regression coefficients. Associated variances, covariances, and intercepts will be automatically determined according to the following rules.

1. The variances of exogenous variables and the residuals of endogenous variables will be set as freely estimated coefficients. The variance coefficients can be classified by the operators `<=>` or `<~>`. For example, the variance of the residual of `y` can be explicitly set as free via `y <=> y`. To penalize the variance, we can use `y <~> y`. Note that it is rare to treat variances as penalized.

2. The covariances among all the exogenous variables will be freely estimated. The covariances can be also classified through `<=>` or `<~>`. For example, the covariance among `x1 - x4` can be set as free by `x1 + x2 + x3 + x4 <=> x1 + x2 + x3 + x4`.

3. The intercepts of all observed responses will be treated as free. The types of intercepts can be declared via a directed operator with intercept variable `1` on the undirected side. For example, the freely estimated intercepts can be specified with `y + x1 + x2 + x3 + x4 <= 1`. Note that once the intercept variable `1` appears in the specified model, the intercept for every endogenous response must be declared. Otherwise, it will be set as zero by default.

lslx always includes a mean structure in the specified model because it helps researchers to interpret the values of estimates under their original scales. Note that the inclusion of the default saturated mean structure will not alter the fitting result made by SEM with only covariance structures.

The specified regression model can be represented in several equivalent ways. For example, it is possible to use reverse operators

```
R> model_reg <- "x1 + x2 => y
+                x3 + x4 ~> y"
```

In **lslx**, all directed operators can be reversed. Another way to declare parameter type is using prefix operators

```
R> model_reg <- "y <= x1 + x2 + pen() * x3 + pen() * x4"
```

or equivalently

```
R> model_reg <- "y <~ free() * x1 + free() * x2 + x3 + x4"
```

Here, `pen()` and `free()` are both prefixes. `pen()` makes the corresponding coefficient penalized and `free()` forces it to be freely estimated. A starting value can be placed inside the parenthesis of a prefix. `fix()` is another important prefix that fixes the coefficient at the specified value in the parenthesis. Note that prefixes can stand on either side of the operand. However, any prefix cannot simultaneously appear on both sides of the operand, which may result in an ambiguous specification.

Now, another example is provided. The example specifies a semi-confirmatory factor model with three latent factors (`visual`, `textual`, and `speed`) and nine responses (`x1 - x9`) accommodating the data of Holzinger and Swineford (1939).

```
R> model_fa <- "visual  :=> x1 + x2 + x3
+               textual :=> x4 + x5 + x6
```

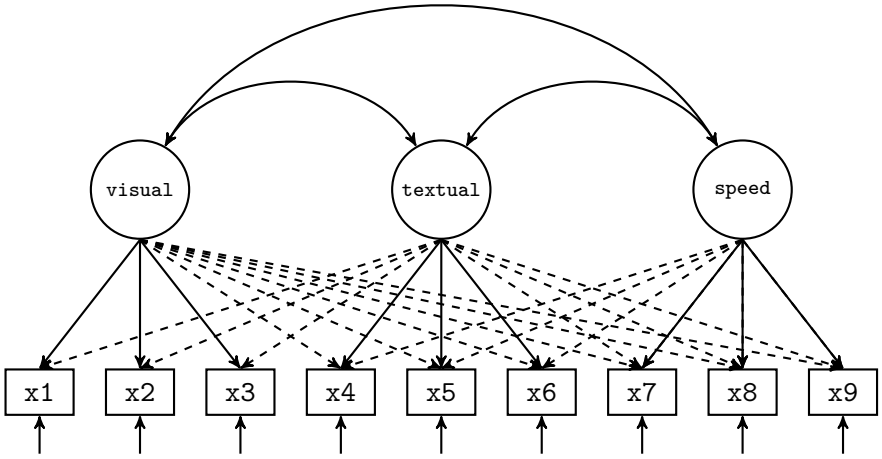

Figure 1: The path diagram of a semi-confirmatory factor analysis model for the data of Holzinger and Swineford (1939). Arrows with solid and broken lines represent freely estimated and penalized parameters, respectively.

```
+               speed   :=> x7 + x8 + x9
+               visual  :~> x4 + x5 + x6 + x7 + x8 + x9
+               textual :~> x1 + x2 + x3 + x7 + x8 + x9
+               speed   :~> x1 + x2 + x3 + x4 + x5 + x6
+               visual  <=> fix(1) * visual
+               textual <=> fix(1) * textual
+               speed   <=> fix(1) * speed"
```

In **lslx**, operators `:=>` and `:~>` (or their reversed counterparts `<=:` and `<~:`) are used to define latent factors. Operator `:=>` sets loadings to be freely estimated and `:~>` makes them penalized. The above factor model is depicted by the path diagram in Figure 1. All factor loadings are estimated. Some of them are freely estimated (solid line arrow) and some of them are penalized (broken line arrow). The specification says that each response variable is mainly an indicator of some latent factor (represented by the freely estimated loadings). However, the possibility of cross-loadings is not excluded (explored through the penalized loadings). The relationships among latent factors can also be specified via `=>`, `~>`, `<=>`, and `<~>` (or their possibly reverse versions) according to the syntax introduced earlier. Because the covariances among exogenous variables will be automatically set as free, they do not have to be explicitly stated here. Only the variances of latent factors are declared as fixed for scale setting. Note that **lslx** will not automatically fix appropriate loadings or variances to set the scales of latent factors. Users must do this job by their own for the purpose of clarity.

lslx also accepts basic **lavaan** operators, including `~`, `=~`, and `~~`. For example, the previous factor model can be equivalently respecified as

```
R> model_fa_lavaan <- "visual  =~ x1 + x2 + x3
+                      textual =~ x4 + x5 + x6
+                      speed   =~ x7 + x8 + x9
```

```
+                       pen() * visual  =~ x4 + x5 + x6 + x7 + x8 + x9
+                       pen() * textual =~ x1 + x2 + x3 + x7 + x8 + x9
+                       pen() * speed   =~ x1 + x2 + x3 + x4 + x5 + x6
+                       visual  ~~ 1 * visual
+                       textual ~~ 1 * textual
+                       speed   ~~ 1 * speed"
```

The model parser automatically replaces **lavaan** operators `~`, `=~`, and `~~` by `<=`, `:=>`, and `<=>`, respectively. In addition, a numeric prefix makes the corresponding parameter to be fixed at the specified value.

4.2. `$new()`, `$fit()`, and `$summarize()` methods

To conduct a semi-confirmatory factor analysis, an `lslx` object can be initialized via the `$new()` method

```
R> lslx_fa <- lslx$new(model = model_fa,
+    data = lavaan::HolzingerSwineford1939)

An 'lslx' R6 class is initialized via 'data' argument.
  Response Variables: x1 x2 x3 x4 x5 x6 x7 x8 x9
  Latent Factors: visual textual speed
```

Here, an `lslx` object called `lslx_fa` is created with the previously specified `model_fa` and the data set `HolzingerSwineford1939` in **lavaan**. The `data` argument must be supplied with a `data.frame` with column names that match the names of the response variables specified in the `model` argument. If an `lslx` object is successfully initialized, the names of the response variables and the latent factors will be displayed by the shell. To suppress the displaying of feedback, one can use `verbose = FALSE`. **lslx** also supports initialization through importing sample moments. In that case, arguments `sample_cov` and `sample_size` must be supplied (possibly `sample_mean`).

To fit the specified model to the imported data, the `$fit()` method can be used

```
R> lslx_fa$fit(penalty_method = "mcp",
+    lambda_grid = seq(.01, .60, .01), delta_grid = c(1.5, 3.0, Inf))

CONGRATS: Algorithm converges under EVERY specified penalty level.
  Specified Tolerance for Convergence: 0.001
  Specified Maximal Number of Iterations: 100
```

The `$fit()` method requires users to mainly specify three arguments: `penalty_function`, `lambda_grid`, and `delta_grid`. The `penalty_function` argument is used to specify the penalty function by values of `"none"`, `"lasso"` (for LASSO), or `"mcp"` (for MCP). The `lambda_grid` and `delta_grid` arguments are designed to set the penalty and convexity levels, respectively. The above sample code implements Algorithm 2 to compute PL estimates with MCP under $\Lambda \times \Delta = \{.01, .02, ..., .60\} \times \{1.5, 3.0, \infty\}$. By default, a message will be printed

to show whether the optimization algorithm converges across all penalty and convexity levels. The fitting results are stored in `lslx_fa` and can be later manipulated by other built-in methods of `lslx` class.

After deriving the fitting results, the easiest way to probe their content is to implement the `$summarize()` method with a selector specified in argument `selector`

```
R> lslx_fa$summarize(selector = "bic", interval = FALSE)

General Information
   number of observations                                301
   number of complete observations                       301
   number of missing patterns                           none
   number of groups                                        1
   number of responses                                     9
   number of factors                                       3
   number of free coefficients                            30
   number of penalized coefficients                       18

Numerical Conditions
   selected lambda                                     0.140
   selected delta                                      1.500
   objective value                                     0.158
   objective gradient absolute maximum                 0.000
   objective Hessian convexity                         0.593
   number of iterations                                4.000
   loss value                                          0.103
   number of non-zero coefficients                    34.000
   degrees of freedom                                 20.000
   robust degrees of freedom                          20.640
   scaling factor                                      1.032

Fit Indices
   root mean square error of approximation (rmsea)     0.043
   comparative fit index (cfi)                         0.988
   non-normed fit index (nnfi)                         0.978
   standardized root mean of residual (srmr)           0.030

Likelihood Ratio Test
                    statistic         df    p-value
   unadjusted          30.919     20.000      0.056
   mean-adjusted       29.961     20.000      0.070

Root Mean Square Error of Approximation Test
                     estimate      lower      upper
   unadjusted           0.043      0.000      0.075
   mean-adjusted        0.041      0.000      0.075
```

```
Coefficient Test (St.Error = "sandwich")
  Factor Loading
                     type  estimate  std.error  z-value  p-value
      x1<-visual     free     0.709      0.095    7.482    0.000
      x2<-visual     free     0.555      0.082    6.799    0.000
      x3<-visual     free     0.744      0.070   10.577    0.000
      x4<-visual      pen     0.000          -        -        -
      x5<-visual      pen    -0.102      0.073   -1.401    0.081
      x6<-visual      pen     0.000          -        -        -
      x7<-visual      pen    -0.255      0.119   -2.151    0.016
      x8<-visual      pen     0.000          -        -        -
      x9<-visual      pen     0.323      0.075    4.287    0.000
     x1<-textual      pen     0.255      0.081    3.136    0.001
     x2<-textual      pen     0.000          -        -        -
     x3<-textual      pen     0.000          -        -        -
     x4<-textual     free     0.987      0.061   16.133    0.000
     x5<-textual     free     1.143      0.061   18.582    0.000
     x6<-textual     free     0.913      0.058   15.727    0.000
     x7<-textual      pen     0.000          -        -        -
     x8<-textual      pen     0.000          -        -        -
     x9<-textual      pen     0.000          -        -        -
       x1<-speed      pen     0.000          -        -        -
       x2<-speed      pen     0.000          -        -        -
       x3<-speed      pen     0.000          -        -        -
       x4<-speed      pen     0.000          -        -        -
       x5<-speed      pen     0.000          -        -        -
       x6<-speed      pen     0.000          -        -        -
       x7<-speed     free     0.825      0.106    7.805    0.000
       x8<-speed     free     0.731      0.069   10.604    0.000
       x9<-speed     free     0.499      0.063    7.962    0.000

  Covariance
                     type  estimate  std.error  z-value  p-value
 textual<->visual    free     0.325      0.086    3.765    0.000
   speed<->visual    free     0.375      0.100    3.731    0.000
  speed<->textual    free     0.278      0.078    3.572    0.000

  Variance
                     type  estimate  std.error  z-value  p-value
  visual<->visual   fixed     1.000          -        -        -
textual<->textual   fixed     1.000          -        -        -
    speed<->speed   fixed     1.000          -        -        -
          x1<->x1    free     0.671      0.113    5.951    0.000
          x2<->x2    free     1.073      0.106   10.165    0.000
          x3<->x3    free     0.720      0.090    7.955    0.000
          x4<->x4    free     0.377      0.050    7.504    0.000
          x5<->x5    free     0.416      0.060    6.875    0.000
```

```
          x6<->x6     free     0.362      0.046    7.902    0.000
          x7<->x7     free     0.595      0.109    5.444    0.000
          x8<->x8     free     0.488      0.084    5.780    0.000
          x9<->x9     free     0.540      0.064    8.431    0.000

Intercept
                      type  estimate  std.error  z-value  p-value
            x1<-1     free     4.936      0.067   73.473    0.000
            x2<-1     free     6.088      0.068   89.855    0.000
            x3<-1     free     2.250      0.065   34.579    0.000
            x4<-1     free     3.061      0.067   45.694    0.000
            x5<-1     free     4.341      0.074   58.452    0.000
            x6<-1     free     2.186      0.063   34.667    0.000
            x7<-1     free     4.186      0.063   66.766    0.000
            x8<-1     free     5.527      0.058   94.854    0.000
            x9<-1     free     5.374      0.058   92.546    0.000
```

The result shown is for fitting the model under $(\hat{\lambda}, \hat{\delta}) = (.14, 1.5)$ selected by BIC. The confidence intervals of the coefficients were not printed because of `interval = FALSE`. The fit indices show that the final model fits the data reasonably well. Despite that 18 penalized loadings were estimated, only four of them were identified as non-zero, including `x5<-visual`, `x7<-visual`, `x9<-visual`, and `x1<-textual`. By default, **lslx** always implements robust statistical inferences provided that the raw data is available. This includes the mean-adjusted LR test (Satorra and Bentler 1994), the mean-adjusted RMSEA (root mean square error of approximation) intervals (Brosseau-Liard, Savalei, and Li 2012; Li and Bentler 2006), and the sandwich standard errors for coefficients (e.g., Browne 1984; Yuan and Hayashi 2006). However, these inference results should be cautiously interpreted because the final model is determined by some selection process.

An interesting property of PL is that a sufficiently large λ can shrink all the penalized parameters to be zero. For example, `lslx_fa$summarize(lambda = 0.6, delta = Inf)` shows that all the penalized loadings are exactly zero, which coincides with the confirmatory factor analysis (CFA) model often used to fit `HolzingerSwineford1939` (e.g., the example of `cfa()` in **lavaan**). In fact, this PL fitting result is numerically identical to the CFA made by **lavaan** (e.g., the same value of LR statistic). On the other hand, a very small λ can yield a result similar to an exploratory factor analysis. This is why the PL can be seen as a methodology bridging the traditional SEM and the exploratory SEM.

4.3. Other build-in methods

In **lslx**, there are several built-in methods to adjust the inner status and manipulate the fitting results of an `lslx` object. These methods are listed in Table 1 and 2. From the viewpoint of data analysis, the set-related methods and plot-related methods are the most relevant.

The set-related methods are designed to alter the initially specified model. To penalize coefficients by name (with given starting values), we may use `$penalize_coefficient()`. In **lslx**, every coefficient name is constructed by variable names appended by `<-`/`<->`, where `<-` and `<->` describe a directed and an undirected effect, respectively. For example, `x1<-visual` is the name for the regression coefficient (or loading) derived from `visual` to `x1`. Note that

Methods	Description
set-related methods	
`$free_coefficient()`	free the specified coefficient
`$penalize_coefficient()`	penalize the specified coefficient
`$fix_coefficient()`	fix the specified coefficient
`$free_directed()`	free the specified directed relation
`$penalize_directed()`	penalize the specified directed relation
`$fix_directed()`	fix the specified directed relation
`$free_undirected()`	free the specified undirected relation
`$penalize_undirected()`	penalize the specified undirected relation
`$fix_undirected()`	fix the specified undirected relation
`$free_block()`	free coefficients in the specified block
`$penalize_block()`	penalize coefficients in the specified block
`$fix_block()`	fix coefficients in the specified block
`$free_heterogeneity()`	free the heterogeneity of the specified block
`$penalize_heterogeneity()`	penalize the heterogeneity of the specified block
`$fix_heterogeneity()`	fix the heterogeneity of the specified block
plot-related methods	
`$plot_numerical_condition()`	plot numerical conditions over $\Lambda \times \Delta$
`$plot_information_criterion()`	plot information criteria over $\Lambda \times \Delta$
`$plot_fit_index()`	plot fit indices over $\Lambda \times \Delta$
`$plot_coefficient()`	plot coefficients over $\Lambda \times \Delta$
test-related methods	
`$test_lr()`	return likelihood ratio (LR) test result
`$test_rmsea()`	return RMSEA interval result
`$test_coefficient()`	return coefficient test result

Table 1: List of set-related, plot-related, and test-related methods in **lslx**. For details, please see the help page of **lslx**.

`<=:/<~:` are not used for naming coefficients and `<-` cannot be reversed. However, penalizing many coefficients by their names may be tedious. Instead, we can use `$penalize_block()` to penalize coefficients by type in a given block. A block is formulated by `y/f/1` and `<-/<->`, where `y` stands for response variables, `f` denotes latent factors, and `1` represents the intercept variable. For example, the block formulated by `y<-1` contains all intercepts of the response variables. In the illustrated examples in Section 5, we will show how to use set-related methods to quickly modify the initially specified model.

The plot-related methods can be used to plot the fitting results stored in an `lslx` object. For example, the `$plot_numerical_condition()` method displays the numerical conditions for assessing the quality of optimization. It can be called by

```
R> lslx_fa$plot_numerical_condition()
```

Figure 2 displays how the number of iterations, the maximum element of absolute sub-gradient, and the minimum diagonal element of approximated Hessian change by penalty level and convexity level. We can see that the algorithm converges within a few iterations under all specified penalty and convexity levels, and there are no non-convexity problems (indicated by

Methods	Description
get-related methods	
`$get_model()`	return deep copy of `model` field
`$get_data()`	return deep copy of `data` field
`$get_fitting()`	return deep copy of `fitting` field
extract-related methods	
`$extract_specification()`	return model specification
`$extract_saturated_mean()`	return saturated sample covariance matrice(s)
`$extract_saturated_cov()`	return saturated sample mean vector(s)
`$extract_saturated_moment_acov()`	return asymptotic covariance matrice(s) of saturated moments
`$extract_lambda_grid()`	return Λ used for fitting
`$extract_delta_grid()`	return Δ used for fitting
`$extract_penalty_level()`	return index name of the optimal penalty level
`$extract_numerical_condition()`	return numerical conditions
`$extract_information_criterion()`	return values of information criteria
`$extract_fit_index()`	return values of fit indices
`$extract_coefficient()`	return estimates of the coefficients
`$extract_implied_cov()`	return model-implied covariance matrice(s)
`$extract_implied_mean()`	return model-implied mean vector(s)
`$extract_residual_cov()`	return residual matrice(s) of covariance
`$extract_residual_mean()`	return residual vector(s) of mean
`$extract_coefficient_matrix()`	return coefficient matrice(s)
`$extract_moment_jacobian()`	return Jacobian of moment structure
`$extract_expected_information()`	return expected Fisher information matrix
`$extract_observed_information()`	return observed Fisher information matrix
`$extract_score_acov()`	return asymptotic covariance of scores
`$extract_coefficient_acov()`	return asymptotic covariance of coefficients
`$extract_loss_gradient()`	return gradient of loss function
`$extract_regularizer_gradient()`	return sub-gradient of regularizer
`$extract_objective_gradient()`	return sub-gradient of objective function

Table 2: List of get-related and extract-related methods in **lslx**. For details, please see the help page of **lslx**.

positive values of all `objective Hessian convexity`). Note that the non-convexity problem is detected via an approximated Hessian used in the optimization algorithm, not the exact one. Another useful plotting method is `$plot_coefficient()`, which draws solution paths. The following code plots the solution paths of coefficients in the block of `y<-f`, i.e., the factor loadings.

```
R> lslx_fa$plot_coefficient(block = "y<-f")
```

In Figure 3, we can observe how the values of loadings change by penalty level and convexity level. Under $\delta = 1.5$, MCP shrinks the values of estimates sharply. On the contrary, infinite δ yields relatively smooth solution paths.

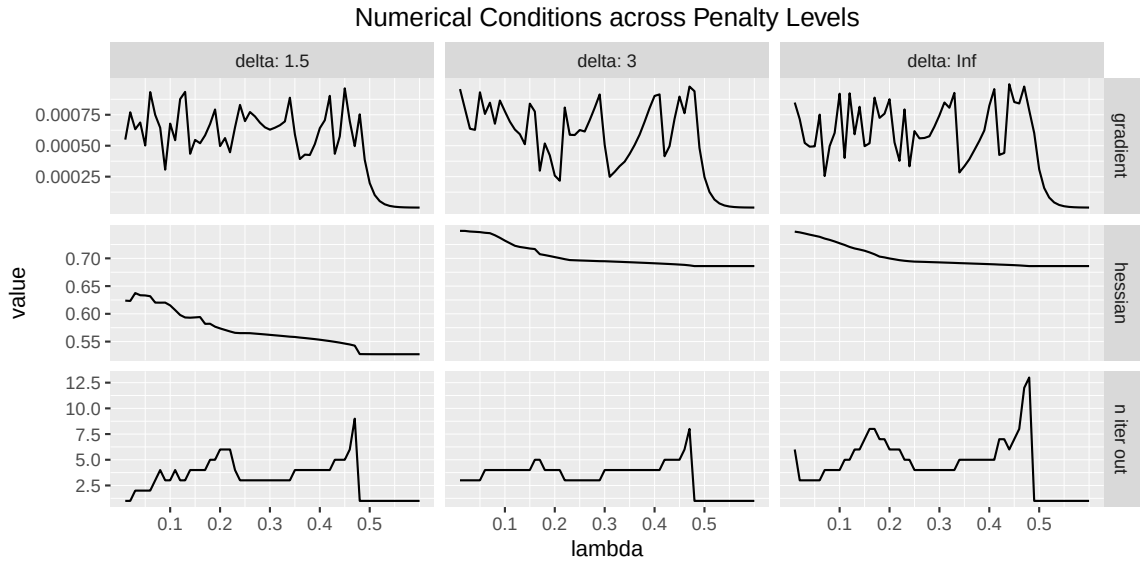

Figure 2: Maximum element of absolute sub-gradient (top), minimum diagonal element of approximated Hessian (middle), and number of iterations (bottom) across all the penalty and convexity levels.

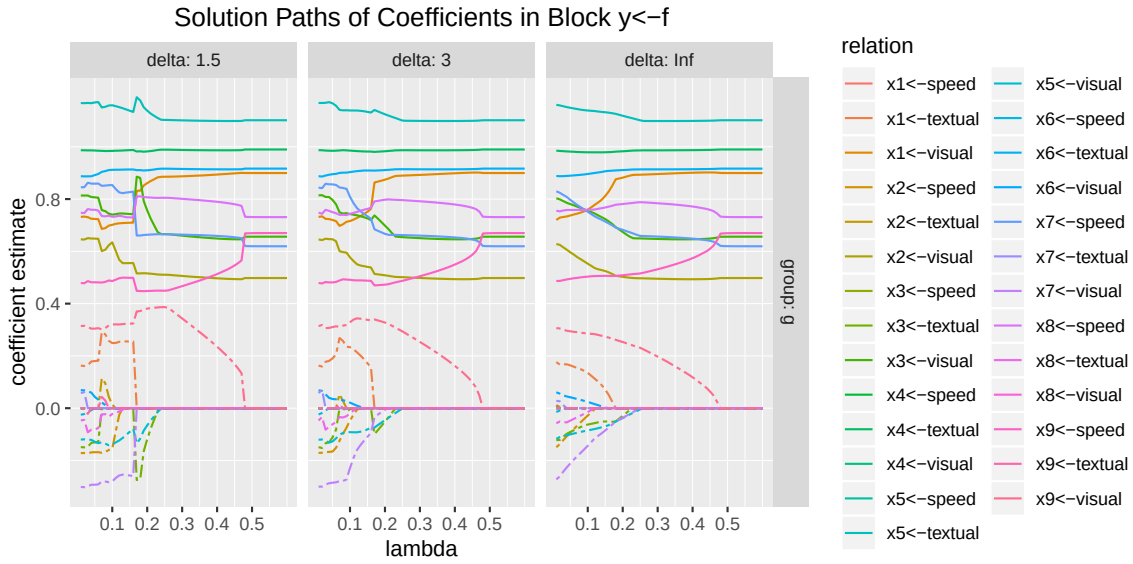

Figure 3: Solution paths of factor loading estimates across all the penalty and convexity levels. Freely estimated and penalized loadings are represented by solid and broken lines, respectively.

4.4. Practical guidelines

This subsection discusses several practical issues when using **lslx**, including the check of model identification, the initialization of Λ and Δ, and the choice of selectors.

The first issue is about model identification. In a semi-confirmatory analysis, the specified

model is sometimes not identified under the usual SEM framework (e.g., `model_fa`). However, PL can still estimate it because the penalty term introduces additional constraints on the penalized parameters. For example, with LASSO the optimization problem in Equation 4 can be equivalently represented by $\min_\theta \mathcal{D}(\theta)$ such that $\sum_{q=1}^{Q} c_{\theta_q}|\theta_q| \leq \gamma$ for some $\gamma > 0$ (see Tibshirani 1996). In fact, PL is often implemented as a solution to overcome the $P > N$ problem (underidentified model) in regression analysis. Despite there is no general rule to ensure the identifiability before PL fitting, it can be checked empirically. Motivated by Shapiro and Browne (1983), the local identifiability of the selected model can be checked by examining whether the smallest singular value of $\frac{\partial \tau(\hat{\theta})}{\partial \vartheta^\top}$ is numerically larger than zero (see also Huang *et al.* 2017), where $\hat{\vartheta}$ is a vector formed by the freely estimated and penalized non-zero elements of $\hat{\theta}$, the so-called effective elements of $\hat{\theta}$. For example, our BIC selected factor model can be checked by

```
R> moment_jacobian <- lslx_fa$extract_moment_jacobian(
+    selector = "bic", type = "effective")
R> min(svd(moment_jacobian)$d)

[1] 0.211
```

Because the value is evidently larger than zero, we conclude that the selected model is at least locally identified. Note that by default `$extract_moment_jacobian()` returns the whole model Jacobian matrix. To only extract the Jacobian with respect to the effective parameters, `type = "effective"` should be used.

The second issue is how to initialize Λ and Δ. Despite that **lslx** automatically initializes them by setting `lambda_grid = "default"` and `delta_grid = "default"`, the discussion below can help users to understand how **lslx** works. In linear regression, Λ is often initialized by (1) setting λ_1 as a small number (e.g., $\lambda_1 \approx \log(N)/N$ in **lslx**) and λ_K as a minimal value that shrinks all the penalized parameters to be zero; (2) constructing K values decreasing from λ_K to λ_1 on the log scale (e.g., Friedman *et al.* 2010). However, making all penalized parameters to be zero is unnecessary in practice. Let t denote a specified upper bound such that λ_K can shrink any standardized and unpenalized $|\tilde{\theta}^*_q| \leq t$ to be zero. Based on the rationale described in Appendix B, λ_K can be loosely approximated by

$$\lambda_K \approx \frac{\sigma_{\max}}{\sigma_{\min}(1 - r^2_{\max})} t, \tag{19}$$

where $\sigma_{\min}$ and $\sigma_{\max}$ are the maximum and minimum standard deviation of both response variables and latent factors, respectively, and $r^2_{\max}$ is the largest coefficient of determination for endogenous variables. For example, by using $t = 0.3$, $r^2_{\max} = 0.6$, and $\sigma_{\max} = \sigma_{\min} = 1$, we have $\lambda_K \approx 0.75$. To initialize Δ, it should be known that a small δ may result in a non-convexity problem and a too large δ suffers from the problem of biased estimation. A loose approximation for δ_1, the smallest element of Δ, is

$$\delta_1 \approx \frac{\sigma^2_{\max}(1 - r^2_{\min})}{\sigma^2_{\min}}, \tag{20}$$

where $r^2_{\min}$ is the smallest coefficient of determination for endogenous variables. The largest element δ_L is often set as infinity to obtain a LASSO solution, which is used as a warm

start for calculating $\hat{\theta}$ under a smaller δ (see Mazumder *et al.* 2011). In practice, too small values of δ often make problematic fitting results (e.g., non-convergence or non-convexity of approximated Hessian matrix). By default, **lslx** will not use them in `$summarize()` or other methods relying on fitting results. To include these problematic results, users should set `include_faulty = TRUE`, but it is generally not recommended.

The third issue is the choice of selectors. The information criteria in **lslx** can be distinguished into three types: AIC-type (AIC and AIC3), BIC-type (BIC and CAIC), and mixed-type (HBIC and ABIC). By their relative orders of C_N, it is expected that BIC-type is the most conservative (i.e., it results in the sparsest estimate with the price of lower goodness-of-fit), followed by mixed-type, and then AIC-type which tends to choose a relatively complex model. In theory, a BIC-type criterion asymptotically choose a quasi-true model with probability one (e.g., Huang 2017a). However, under small sample sizes or weak signals, an AIC-type criterion generally yields better selection results (e.g., Vrieze 2012). The behavior of a mixed-type criterion is also mixed. It performs close to AIC under small sample sizes and becomes similar to BIC asymptotically. Despite a mixed-type criterion might not outperform its competitors in the home field of AIC or BIC (e.g., small sample size or strong signal settings), its overall performance across different conditions is good (e.g., Lin, Huang, and Weng 2017). If users don't have strong arguments to use an AIC-type or a BIC-type criterion, the author would recommend employing a mixed-type one.

5. Advanced functionality

In this section, two advanced functions of **lslx** are described: the two-stage method with auxiliary variables for handling missing data and the reparameterized multi-group SEM to explore population heterogeneity.

5.1. Two-stage method for missing data

When conducting SEM, it is likely to encounter missing values. In **lslx**, missing values can be handled by the listwise deletion (LD) method and the two-stage (TS) method (Yuan and Bentler 2000). LD only uses complete observations for further analysis. If the mechanism is missing completely at random (MCAR; Rubin 1976), LD can yield a consistent estimator. However, LD suffers from loss of efficiency because the dropped incomplete cases still carry information for estimation. On the other hand, TS first minimizes the likelihood based on all available observations to calculate saturated moments. The obtained moment estimates are used for subsequent SEM analysis. Under the assumption of missing at random (MAR; Rubin 1976), TS is consistent. In addition, the standard errors of coefficients can be consistently estimated (e.g., Yuan and Lu 2008). Compared with LD, TS is generally valid (in terms of consistency) and more efficient (with respect to mean squared error), thus **lslx** sets TS as the default. The current version also supports the inclusion of auxiliary variables to make the MAR assumption more plausible (Savalei and Bentler 2009).

Specifically, let $\mathcal{Y}^o = \{y_n^o\}_{n=1}^N$ denote an observed random sample with y_n^o being the observed part of y_n. The first stage of TS aims to estimate the saturated mean $\mu(\tau)$ and covariance matrix $\Sigma(\tau)$ based on $\mathcal{Y}^o$, where $\tau^\top = (\mu^\top, \sigma^\top)$ is a saturated moment vector with $\sigma =$

vech(Σ). To obtain $\hat{\tau}$, the TS method maximizes the following likelihood function

$$\mathcal{L}(\tau) = -\frac{1}{2N}\sum_{n=1}^{N}\log|\Sigma_n(\tau)| - \frac{1}{2N}\sum_{n=1}^{N}[y_n^o - \mu_n(\tau)]^\top\Sigma_n(\tau)^{-1}[y_n^o - \mu_n(\tau)], \tag{21}$$

where $\mu_n(\tau)$ and $\Sigma_n(\tau)$ are the saturated mean and covariance structure for y_n^o. Equation 21 can be optimized by using an expectation-maximization (EM) algorithm (Dempster, Laird, and Rubin 1977). In the second stage of TS, Equation 4 is solved using Algorithm 2 with m and S replaced by $\mu(\hat{\tau})$ and $\Sigma(\hat{\tau})$, respectively.

One may ask why **lslx** doesn't implement the so-called full-information (FI) approach to handle missing values (Enders and Bandalos 2001). The main reason is that the TS method is efficient in terms of computation time. The additional cost introduced by TS is only for calculating $\hat{\tau}$ in the first-stage. In contrast, the FI approach requires an expectation step before each outer iteration (Jamshidian and Bentler 1999). Additionally, current evidence shows that the FI approach has no particular advantage over TS, both theoretically (Yuan and Bentler 2000) and empirically (Savalei and Bentler 2009; Savalei and Falk 2014).

Now, we demonstrate how to use **lslx** to conduct TS using the data from Holzinger and Swineford (1939) again. Because the original data set is complete, missing values are created according to the example of `twostage()` in package **semTools** (semTools Contributors 2016). Missingness in `x5` and `x9` depends on the values of `x1` and `age`, respectively.

```
R> data_miss <- lavaan::HolzingerSwineford1939
R> data_miss$x5 <- ifelse(
+    test = data_miss$x1 <= quantile(data_miss$x1, .3),
+    yes = NA, no = data_miss$x5)
R> data_miss$age <- data_miss$ageyr + data_miss$agemo / 12
R> data_miss$x9 <- ifelse(
+    test = data_miss$age <= quantile(data_miss$age, .3),
+    yes = NA, no = data_miss$x9)
```

An `lslx` object is initialized with a relatively parsimonious CFA model. To include auxiliary variables, the `auxiliary_variable` argument should be declared.

```
R> model_miss <- "x1 + x2 + x3 <=: visual
+                 x4 + x5 + x6 <=: textual
+                 x7 + x8 + x9 <=: speed
+                 visual  <=> 1 * visual
+                 textual <=> 1 * textual
+                 speed   <=> 1 * speed"
R> lslx_miss <- lslx$new(model = model_miss, data = data_miss,
+    auxiliary_variable = c("ageyr", "agemo"), verbose = FALSE)
```

In this example, `"ageyr"` and `"agemo"` are set as auxiliary variables. Since the CFA model may not fit the data well due to its independent cluster structure for loadings, motivated by Bayesian SEM (Muthén and Asparouhov 2012), a correlated residuals structure is considered. In fact, PL can be also seen as a maximum of a posteriori (MAP) method under a Bayesian framework (see Meng 2008; Strawderman, Wells, and Schifano 2013). The `$penalize_block()` method can be used to penalize coefficients in a specified block by type

```
R> lslx_miss$penalize_block(block = "y<->y", type = "fixed", verbose = FALSE)
```

This code sets all coefficients with `type = "fixed"` in `block = "y<->y"` as penalized. Despite that such a model is not identified under the usual SEM framework, PL can still estimate it. The CFA model with correlated residuals is fitted to the data via the `$fit_lasso()` method, which is a convenient wrapper for `$fit()` with `penalty_method = "lasso"`

```
R> lslx_miss$fit_lasso(verbose = FALSE)
```

By default, **lslx** implements the TS method for handling missing data. It can be explicitly set by `missing_method = "two_stage"`. If any auxiliary variable is specified when initializing an `lslx` object, the variable will be included to estimate the saturated moments. Finally, the robust AIC is utilized to select an optimal penalty level

```
R> lslx_miss$summarize(selector = "raic", style = "minimal")
```

```
General Information
   number of observations                301.000
   number of complete observations       138.000
   number of missing patterns              4.000
   number of groups                        1.000
   number of responses                     9.000
   number of factors                       3.000
   number of free coefficients            30.000
   number of penalized coefficients       36.000

Numerical Conditions
   selected lambda                         0.130
   selected delta                           none
   objective value                         0.168
   objective gradient absolute maximum     0.001
   objective Hessian convexity             0.741
   number of iterations                    2.000
   loss value                              0.051
   number of non-zero coefficients        41.000
   degrees of freedom                     13.000
   robust degrees of freedom              16.173
   scaling factor                          1.244
```

From the summary with `style = "minimal"`, we can see that 11 of the 36 penalized coefficients are identified as non-zero. To see which residual covariances are non-zero, the `$extract_coefficient_matrix()` method can be used.

```
R> lslx_miss$extract_coefficient_matrix(selector = "raic", block = "y<->y")
```

```
$g
       x1     x2      x3     x4      x5      x6      x7     x8      x9
```

```
x1  0.4735  0.000  0.0000 0.0485 -0.1116  0.0000 -0.0801 0.0000  0.0000
x2  0.0000  1.150  0.1034 0.0000  0.0000  0.0000 -0.1050 0.0000  0.0000
x3  0.0000  0.103  0.8780 0.0000 -0.0791  0.0000  0.0000 0.0000  0.0000
x4  0.0485  0.000  0.0000 0.3875  0.0000  0.0000  0.0793 0.0000  0.0000
x5 -0.1116  0.000 -0.0791 0.0000  0.4103  0.0000  0.0000 0.0000  0.0079
x6  0.0000  0.000  0.0000 0.0000  0.0000  0.3476  0.0000 0.0125 -0.0170
x7 -0.0801 -0.105  0.0000 0.0793  0.0000  0.0000  0.9302 0.2534  0.0000
x8  0.0000  0.000  0.0000 0.0000  0.0000  0.0125  0.2534 0.7189  0.0000
x9  0.0000  0.000  0.0000 0.0000  0.0079 -0.0170  0.0000 0.0000  0.3348
```

The values of fit indices show that this final model with many residual covariances fits the data very well.

```
R> lslx_miss$extract_fit_index(selector = "raic")
```

```
 rmsea    cfi   nnfi   srmr
0.0246 0.9972 0.9923 0.0315
```

In general, the author doesn't recommend the use of the correlated residuals structure because the specified model is too exploratory and the resulting model may be difficult to interpret.

5.2. Multi-group SEM analysis

Multi-group SEM (MGSEM) is often used to examine the heterogeneity of model coefficients across several populations (Jöreskog 1971; Sörbom 1974). Suppose there are G populations sharing common moment structures $\mu(\cdot)$ and $\Sigma(\cdot)$ but having possibly different values for their model parameter θ_g. Let θ_{gq} denote the q^{th} component of θ_g. Without loss of generality, we assume that θ_{1q}, θ_{2q}, ..., θ_{Gq} represent the same element selected from α, B, or Φ. Hence, a coefficient θ_{gq} is said to be homogeneous across the G populations if $\theta_{1q} = \theta_{2q} = ... = \theta_{Gq}$. Otherwise, we call θ_{gq} heterogeneous.

Given a multi-group random sample $\mathcal{Y} = \{\{y_{gn}\}_{n=1}^{N_g}\}_{g=1}^{G}$, ML estimation tries to obtain $\hat{\theta}_1$, $\hat{\theta}_2$, ..., $\hat{\theta}_G$ by minimizing the following multi-group loss function

$$\begin{aligned}\mathcal{D}(\theta) = &\sum_{g=1}^{G} w_g \left[\mathrm{tr}(S_g\Sigma(\theta_g)^{-1}) - \log|S_g\Sigma(\theta_g)^{-1}| - P\right] \\ &+ \sum_{g=1}^{G} w_g(m_g - \mu(\theta_g))^\top\Sigma(\theta_g)^{-1}(m_g - \mu(\theta_g)),\end{aligned} \tag{22}$$

where $m_g = \frac{1}{N_g}\sum_{n=1}^{N_g} y_{gn}$, $S_g = \frac{1}{N_g}\sum_{n=1}^{N_g}(y_{gn} - m_g)(y_{gn} - m_g)^\top$, and $w_g = \frac{N_g}{N}$ with $N = \sum_{g=1}^{G} N_g$. To test the homogeneity/heterogeneity of parameters across groups, users may impose equality constraints on coefficients. To evaluate the appropriateness of the constraints, they can perform formal statistical tests or use goodness-of-fit indices.

The current version of **lslx** cannot impose equality constraints on the model parameters. Therefore, it may seem that it is incapable of examining coefficient homogeneity/heterogeneity.

However, by using a reparameterized MGSEM with penalization, **lslx** can still explore homogeneity/heterogeneity patterns (see Huang in press). In **lslx**, each group parameter θ_g is parameterized as

$$\theta_g = \underline{\theta} + \underline{\theta}_g, \tag{23}$$

where $\underline{\theta}$ is called the reference component and $\underline{\theta}_g$ is called the increment component of group g. The meaning of $\underline{\theta}$ and $\underline{\theta}_g$ relies on the choice of the reference group. When there is no reference group, i.e., $\underline{\theta} = 0$, θ_g and $\underline{\theta}_g$ are equivalent. If group j is set as reference, $\underline{\theta}_j$ will be set as zero and $\underline{\theta}_g$ will represent the difference of θ_g and θ_j, i.e., $\underline{\theta}_g = \theta_g - \theta_j$. Under this setting, θ_{gq} is homogeneous if and only if $\underline{\theta}_{gq} = 0$ for $g = 1, 2, ..., G$ $(g \neq j)$. Therefore, we can examine the homogeneity/heterogeneity of θ_{gq} by exploring the sparsity pattern of $\underline{\theta}_{1q}$, $\underline{\theta}_{2q}$, ..., $\underline{\theta}_{Gq}$.

In **lslx**, MGSEM analysis is implemented by minimizing a PL criterion composed by the multi-group loss function in Equation 22 and a penalty term as follows

$$\mathcal{R}(\theta, \lambda) = \sum_{q=1}^{Q} c_{\underline{\theta}_q} \rho(|\underline{\theta}_q|, \lambda) + \sum_{g=1}^{G} \sum_{q=1}^{Q} c_{\underline{\theta}_{gq}} \rho(|\underline{\theta}_{gq}|, \lambda). \tag{24}$$

This multi-group optimization problem can also be solved by Algorithm 2. After that, the PL estimates over $\Lambda \times \Delta$ are derived, and an optimal pair of (λ, δ) can be chosen by minimizing some information criterion in Equation 7. In general, the implementation of semi-confirmatory MGSEM is similar to the single-group case except for the emphasis on the homogeneity/heterogeneity of the coefficients across groups.

The following example shows how to use **lslx** to examine strong factorial invariance via MCP. A possible initialization for this purpose is

```
R> model_mgfa <- "1 * x1 + x2 + x3 <=: visual
+                 1 * x4 + x5 + x6 <=: textual
+                 1 * x7 + x8 + x9 <=: speed"
R> lslx_mgfa <- lslx$new(model = model_mgfa,
+    data = lavaan::HolzingerSwineford1939, group_variable = "school",
+    reference_group = "Pasteur", verbose = FALSE)
```

For simplicity, a commonly used independent cluster structure (i.e., each response is only influenced by one latent factor) is considered here. The model fixes the loadings of `x1`, `x4`, and `x7` at one for scale setting. Argument `group_variable` specifies which variable should be used as group label, and `reference_group` determines the reference group. Note that since `"Pasteur"` is set as the reference group, model parameters in `"Grant-White"` are now increment components for representing differences. If argument `reference_group` is missing, the reference component will be set to zero, which is equivalent to the usual parameterization of MGSEM. By default, **lslx** treats all non-trivial model parameters as heterogeneous.

The syntax for specifying a multi-group model is generally similar to that of the single-group model, except that a vectorized prefix can be used. An explicit way to specify the multi-group model is

```
R> model_mgfa <- "c(fix(0), fix(1)) * x1 + x2 + x3 <=: visual
+                 c(fix(0), fix(1)) * x4 + x5 + x6 <=: textual
+                 c(fix(0), fix(1)) * x7 + x8 + x9 <=: speed"
```

Here, `c(fix(0), fix(1))` is a vectorized prefix that sets the corresponding coefficients to zero and one, respectively. In this example, the first group is `"Grant-White"` and the second is `"Pasteur"`. This order corresponds to the `sort()` result for the group names. Note that the coefficients are set to 1 in `"Pasteur"` since `"Pasteur"` is the reference group. For `"Grant-White"`, the increment component should be restricted to 0 to make the corresponding loadings equal to 1 as desired. If `c(fix(0), fix(1))` is replaced by `fix(1)`, the **lslx** parser will still interpret `fix(1)` as `c(fix(0), fix(1))`. Of course, if the `reference_group` argument is not specified, `fix(1)` will be interpreted as `c(fix(1), fix(1))`, i.e., two corresponding loadings will be set to 1.

So far, the model specification is still not complete. A measurement satisfies the condition of strong factorial invariance if all loadings and intercepts are homogeneous across the considered groups (Meredith 1993). By default, **lslx** freely estimates all increment components in `"Grant-White"`. To penalize some of them, we can use the `$penalize_heterogeneity()` method

```
R> lslx_mgfa$penalize_heterogeneity(block = c("y<-f", "y<-1"),
+    group = "Grant-White", verbose = FALSE)
```

The code penalizes every coefficient belonging to block `"y<-f"` and `"y<-1"` in `"Grant-White"` according to its reference component. Since restrictions for loadings and intercepts are imposed, the intercepts of latent factors in `"Grant-White"` can be safely estimated

```
R> lslx_mgfa$free_block(block = "f<-1",
+    group = "Grant-White", verbose = FALSE)
```

To understand how to impose minimal constraints for identification in multi-group factor analysis, refer to Millsap (2011). The specified model is now fitted with MCP through the `$fit_mcp()` method, a wrapper for `$fit()` with `penalty_method = "mcp"`

```
R> lslx_mgfa$fit_mcp(verbose = FALSE)
```

Finally, we display the values of loadings and intercepts to evaluate whether they are invariant under the penalty and convexity level selected by HBIC

```
R> loading <- lslx_mgfa$extract_coefficient_matrix(
+    selector = "hbic", block = "y<-f")
R> intercept <- lslx_mgfa$extract_coefficient_matrix(
+    selector = "hbic", block = "y<-1")
R> loading$"Grant-White" - loading$"Pasteur"
```

```
   visual textual speed
x1      0       0     0
x2      0       0     0
x3      0       0     0
x4      0       0     0
x5      0       0     0
x6      0       0     0
```

```
x7      0       0     0
x8      0       0     0
x9      0       0     0
```

```
R> t(intercept$"Grant-White" - intercept$"Pasteur")
```

```
  x1 x2    x3 x4 x5 x6    x7 x8 x9
1  0  0 -0.53  0  0  0 -0.44  0  0
```

The result shows that the loadings are invariant across the two schools. However, the intercepts of `x3` and `x7` are different. We conclude that the condition of strong factorial invariance is violated. The measurement only satisfies the condition of weak factorial invariance (Meredith 1993), i.e., only loadings are homogeneous across the two schools.

6. Numerical comparison with lsl and regsem

In this section, a numerical comparison between **lslx** (ver.0.6.1), **lsl** (ver.0.5.6), and **regsem** (ver.0.9.2) is reported. So far, no studies have strictly evaluated whether existing packages can reliably find PL estimates for SEM. To this end, the minimum function values obtained by the three packages are compared. In addition, the number of iterations and the computation time are also evaluated to understand the performance aspects of existing algorithms.

A multiple indicators and multiple causes (MIMIC) model (Jöreskog and Goldberger 1975) with nine indicators (y_1 - y_9), three latent factors (f_1 - f_3), and six causes (x_1 - x_6) is considered. The population model for generating data is presented in Figure 4. Data are generated from a multivariate normal distribution with zero mean and the covariance implied by the model.

The numerical comparison is made with different sample sizes (200, 400, 600, and 800) and model specifications (simple and complex). For the simple case, the measurement model is assumed to satisfy an independent cluster structure (i.e., `y2<-f3`, `y5<-f1`, and `y8<-f2` are omitted) with `y1`, `y4`, and `y7` set as anchors. The regression coefficients from causes (`x1 - x6`) to factors (`f1 - f3`) are all estimated with penalty. Other parameters are set as free or fixed according to the sparsity pattern in Figure 4. For the complex case, the model specification is similar to that of the simple one except that all loadings in the non-independent cluster part are now estimated with penalization.

The following five implementations are evaluated: the Fisher scoring (`lslx-fisher`) and the Broyden-Fletcher-Goldfarb-Shanno (`lslx-bfgs`) from **lslx**, the expectation-conditional maximization (`lsl-ecm`) from **lsl**, the so-called default (`regsem-default`) and the coordinate descent (`regsem-cd`) from **regsem**. The LASSO penalty is implemented with fixed penalty level $\lambda = 0.1$. The maximal number of outer and inner (if needed) iterations is 1000 and 50, respectively. The tolerance is specified as 10^{-5}, although these packages utilize different rules for assessing convergence. In each condition, the number of replications is set to 500.

The probabilities of non-convergent samples were first evaluated. Both **lslx** and **lsl** yielded nearly minimal non-convergence probabilities. Across all conditions, the probabilities for `lslx-fisher` and `lslx-bfgs` were 0% - 0.4% and 0% - 1.2%, respectively. For each condition, `lsl-ecm` yielded a perfect convergence rate. On the other hand, **regsem** produced several

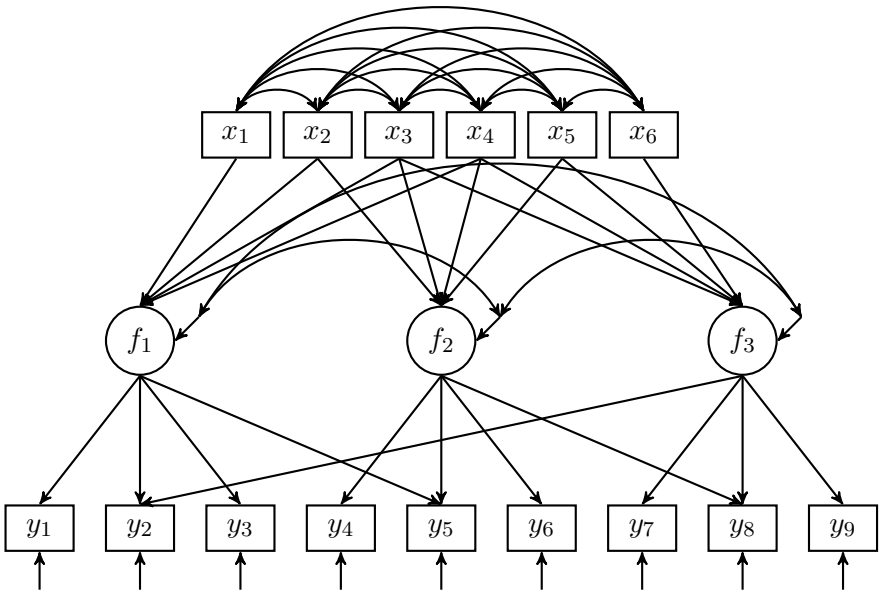

Figure 4: The population MIMIC model for generating data includes nine indicators (y_1 - y_9), three latent factors (f_1 - f_3), and six causes (x_1 - x_6). The loadings in the independent cluster part are set to 0.7. Other non-zero loadings are specified as 0.2. The covariances of the residuals of latent factors and the regression coefficients are all set to 0.2. The covariance among causes is specified as 0.3. The values of the residual variances are so chosen that indicators, factors, and causes are all standardized.

incomplete results. The non-convergence probability of `regsem-cd` was acceptable between 0.8% and 13.2%. However, the probability for `regsem-default` was between 43.4% and 95.6%. We decided to drop `regsem-default` from the rest of the evaluations.

Figure 5 shows the comparison results for each condition based on 500 successful replications (with respect to the remaining four algorithms). The minimum function values indicated that **lslx** and **lsl** yield similar optimization results. It is interesting to note that the two packages implement conceptually different algorithms. Thus, their consistency cannot be explained by the similarity of the underlying algorithms. However, **regsem** always yielded larger function values than **lslx** and **lsl**.

As for the number of iterations and the computation time, `lslx-fisher` and `lslx-bfgs` performed equally well, `lsl-ecm` was slower with more iterations, and `regsem-cd` was the slowest with the most iterations. Note that the difference in computation time cannot be attributed purely to the nature of algorithms, since the computation cores of **lsl** and **regsem** could be speeded up by using compiled code. We observed that `lslx-fisher` requires fewer iterations than `lslx-bfgs`, but spends similar time to achieve convergence. This is likely due to the difference between the more exact but tedious expected Hessian and the less accurate but simpler BFGS Hessian.

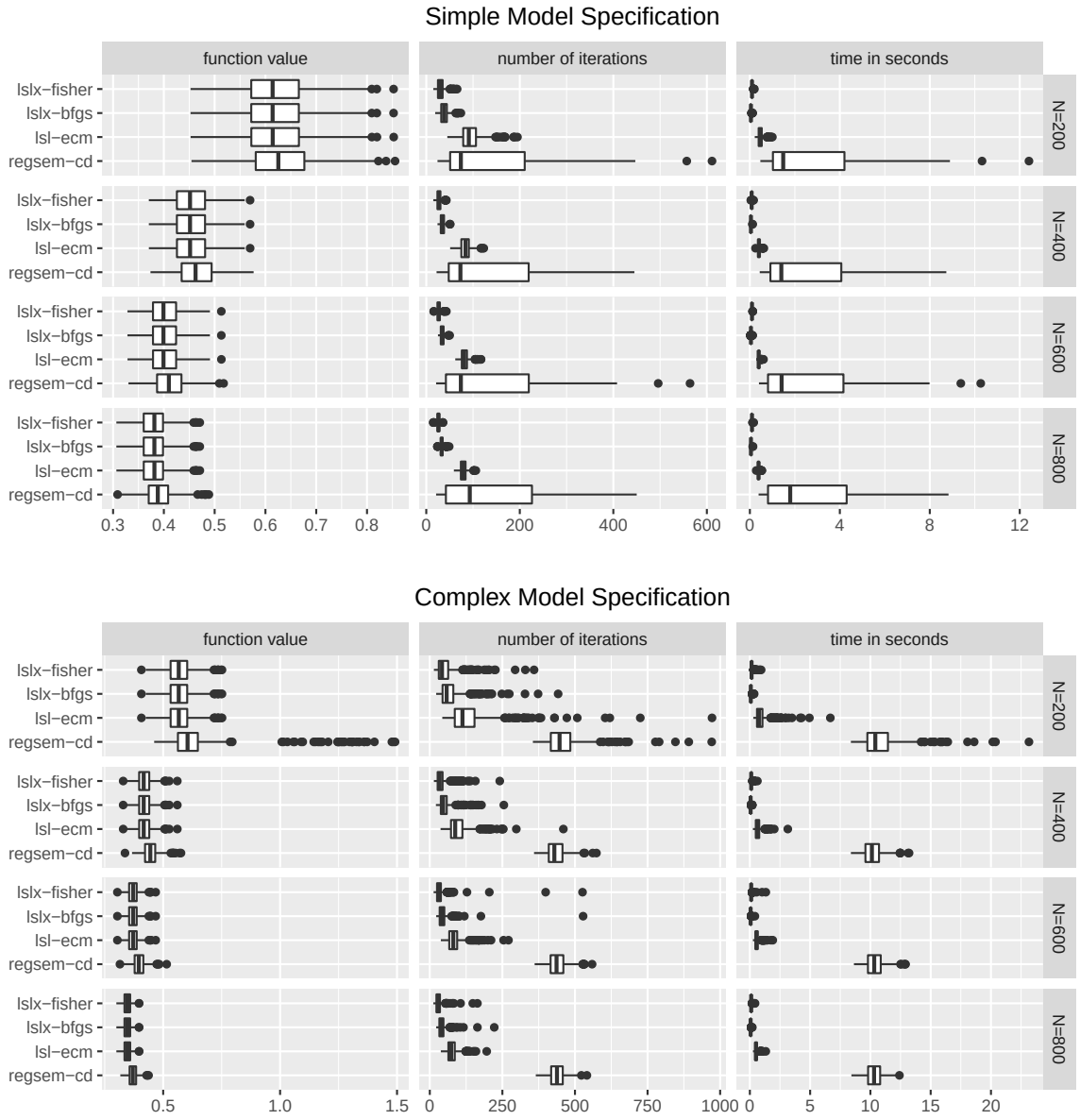

Figure 5: Boxplots of minimum function values, number of iterations, and computation times in seconds for the four algorithms under different sample sizes (200, 400, 600, and 800) and model specifications (simple and complex). The four algorithms are Fisher scoring, Broyden-Fletcher-Goldfarb-Shanno (BFGS), expectation-conditional maximization (ECM), and coordinate descent (CD).

7. Conclusions

In this work, an R package **lslx** is described for semi-confirmatory structural equation modeling (SEM) via penalized likelihood (PL). The package implements a quasi-Newton method to optimize the PL criterion with either LASSO or MCP. To ensure the optimality of the obtained solution, the algorithm checks the first-order condition in each outer iteration. A numerical

comparison between competing packages shows that **lslx** can reliably and efficiently find PL estimates.

Package **lslx** adopts a **lavaan**-like syntax for model specification. The current version also offers an S3 interface for using `lslx` objects, including a wrapper function `plsem()` and related S3 methods. The author believes that most **lavaan** users can easily learn to use **lslx**. The semi-confirmatory SEM is most appropriate when limited substantive theory is available for model specification. Although **lslx** is not the first package for SEM with PL, it is probably the most sophiscated one in terms of usability, dependability, efficiency, and functionality.

Even though the current version of **lslx** can fit a wide class of SEM models, there are still limitations. (1) **lslx** cannot impose linear or non-linear constraints for model parameters. It is worth modifying the current algorithm to incorporate parameter constraints. (2) **lslx** can only handle an ordinal response by treating it as continuous. However, such an approach is only valid under limited conditions (e.g., Rhemtulla, Brosseau-Liard, and Savalei 2012). Implementing a natural PL method for ordinal SEM could enhance the applicability of the current package. (3) **lslx** utilizes inference methods assuming that no model selection has been conducted, which may result in inflated Type I error or confidence interval undercoverage. Recent advances in post-selection inference allow making valid inferences even after model selection (e.g., Berk, Brown, Buja, Zhang, and Zhao 2013; Lee, Sun, Sun, and Taylor 2016). Future versions of **lslx** should implement these methods to control the proportion of false positive findings.

Acknowledgments

The research was supported in part by Grant MOST 104-2410-H-006-119-MY2 from the Ministry of Science and Technology in Taiwan. The author would like to thank Wen-Hsin Hu for preparing the manuscript.

References

Akaike H (1974). "A New Look at the Statistical Model Identification." *IEEE Transactionson Automatic Control*, **19**(6), 716–723. doi:10.1109/TAC.1974.1100705.

Asparouhov T, Muthén B (2009). "Exploratory Structural Equation Modeling." *Structural Equation Modeling: A Multidisciplinary Journal*, **16**(3), 397–438. doi:10.1080/10705510903008204.

Bates D, Eddelbuettel D (2013). "Fast and Elegant Numerical Linear Algebra Using the **RcppEigen** Package." *Journal of Statistical Software*, **52**(5), 1–24. doi:10.18637/jss.v052.i05.

Bentler PM (2006). *EQS 6 Structural Equations Program Manual*. Encino, CA.

Bentler PM, Mooijaart A (1989). "Choice of Structural Model via Parsimony: A Rationale Based on Precision." *Psychological Bulletin*, **106**(2), 315–7. doi:10.1037/0033-2909.106.2.315.

Bentler PM, Weeks DG (1980). "Linear Structural Equations with Latent Variabless." *Psychometrika*, **45**(3), 289–308. doi:10.1007/BF02293905.

Berk R, Brown L, Buja A, Zhang K, Zhao L (2013). "Valid Post-selection Inference." *The Annals of Statistics*, **41**(2), 802–837. doi:10.1214/12-AOS1077.

Bozdogan H (1987). "Model Selection and Akaike's Information Criterion (AIC): The General Theory and Its Analytical Extensions." *Psychometrika*, **52**(3), 345–370. doi:10.1007/BF02294361.

Brosseau-Liard PE, Savalei V, Li L (2012). "An Investigation of the Sample Performance of Two Nonnormality Corrections for RMSEA." *Multivariate Behavioral Research*, **47**(6), 904–930. doi:10.1080/00273171.2012.715252.

Browne MW (1984). "Asymptotically Distribution-Free Methods for the Analysis of Covariance Structures." *British Journal of Mathematical and Statistical Psychology*, **37**(1), 62–83. doi:10.1111/j.2044-8317.1984.tb00789.x.

Cai X, Bazerque JA, Giannakis GB (2013). "Inference of Gene Regulatory Networks with Sparse Structural Equation Models Exploiting Genetic Perturbations." *PLoS Computational Biology*, **9**(5), e1003068. doi:10.1371/journal.pcbi.1003068.

Chang W (2017). ***R6**: Classes with Reference Semantics*. URL https://cran.r-project.org/package=R6.

Chen Y, Liu J, Xu G, Ying Z (2015). "Statistical Analysis of Q-Matrix Based Diagnostic Classification Models." *Journal of the American Statistical Association*, **110**(510), 850–866. ISSN 0162-1459. doi:10.1080/01621459.2014.934827.

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), 1–38.

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.

Enders CK, Bandalos DL (2001). "The Relative Performance of Full Information Maximum Likelihood Estimation for Missing Data in Structural Equation Models." *Structural Equation Modeling*, **8**(3), 430–457. doi:10.1207/S15328007SEM0803_5.

Epskamp S (2017). ***lvnet**: Latent Variable Network Modeling*. URL https://CRAN.R-project.org/package=lvnet.

Epskamp S, Rhemtulla M, Borsboom D (2017). "Generalized Network Psychometrics: Combining Network and Latent Variable Models." *Psychometrika*, **82**(4), 904–927. doi:10.1007/s11336-017-9557-x.

Fan J, Li R (2001). "Variable Selection via Nonconcave Penalized Likelihood and its Oracle Properties." *Journal of the American Statistical Association*, **96**(456), 1348–1360. doi:10.1198/016214501753382273.

Fan RE, Chang KW, Hsieh CJ, Wang XR, Lin CJ (2008). "**LIBLINEAR**: A Library for Large Linear Classification." *Journal of Machine Learning Research*, **9**, 1871–1874.

Fox J, Nie Z, Byrnes J, Culbertson M, DebRoy S, Friendly M, Goodrich B, Jones RH, Kramer A, Monette G, R-Core (2017). ***sem**: Structural Equation Models*. URL https://cran.r-project.org/package=sem.

Friedman J, Hastie T, Tibshirani R (2010). "Regularization Paths for Generalized Linear Models via Coordinate Descent." *Journal of Statistical Software*, **33**(1), 1–22. doi:10.18637/jss.v033.i01.

Hastie T, Tibshirani R, Wainwright M (2015). *Statistical Learning with Sparsity: The Lasso and Generalizations*. Chapman & Hall/CRC.

Haughton DMA (1988). "On the Choice of a Model to Fit Data from an Exponential Family." *The Annals of Statistics*, **16**(1), 342–355. doi:10.1214/aos/1176350709.

Hershberger SL (2003). "The Growth of Structural Equation Modeling: 1994-2001." *Structural Equation Modeling: A Multidisciplinary Journal*, **10**(1), 35–46. doi:10.1207/S15328007SEM1001_2.

Hirose K, Yamamoto M (2014). "Estimation of an Oblique Structure via Penalized Likelihood Factor Analysis." *Computational Statistics and Data Analysis*, **79**, 120–132. doi:10.1016/j.csda.2014.05.011.

Hirose K, Yamamoto M (2015). "Sparse Estimation via Nonconcave Penalized Likelihood in Factor Analysis Model." *Statistics and Computing*, **25**(5), 863–875. doi:10.1007/s11222-014-9458-0.

Holzinger KJ, Swineford F (1939). *A Study in Factor Analysis: The Stability of a Bi-factor Solution*. University of Chicago Press, Chicago.

Huang A (2014). ***sparseSEM**: Sparse-aware Maximum Likelihood for Structural Equation Models*. URL https://cran.r-project.org/package=sparseSEM.

Huang PH (2017a). "Asymptotics of AIC, BIC, and RMSEA for Model Selection in Structural Equation Modeling." *Psychometrika*, **82**(2), 407–426. doi:10.1007/s11336-017-9572-y.

Huang PH (2017b). ***lsl**: Latent Structure Learning*. URL https://cran.r-project.org/package=lsl.

Huang PH (in press). "A Penalized Likelihood Method for Multi-Group Structural Equation Modeling." *British Journal of Mathematical and Statistical Psychology*. doi:10.1111/bmsp.12130.

Huang PH, Chen H, Weng LJ (2017). "A Penalized Likelihood Method for Structural Equation Modeling." *Psychometrika*, **82**(2), 329–354. doi:10.1007/s11336-017-9566-9.

Jacobucci R, Grimm KJ, Brandmaier AM, Serang S (2017). ***regsem**: Regularized Structural Equation Modeling*. URL https://cran.r-project.org/package=regsem.

Jacobucci R, Grimm KJ, McArdle JJ (2016). "Regularized Structural Equation Modeling." *Structure Equation Modeling*, **23**(4), 555–566. doi:10.1080/10705511.2016.1154793.

Jamshidian M, Bentler PM (1999). "ML Estimation of Mean and Covariance Structures with Missing Data Using Complete Data Routines." *Journal of Educational and Behavioral Statistics*, **24**(1), 21–24. doi:10.3102/10769986024001021.

Jöreskog K, Sörbom D (2015). *LISREL 9.20 for Windows*. Skokie, IL.

Jöreskog KG (1971). "Simultaneous Factor Analysis in Several Populations." *Psychometrika*, **36**(4), 409–426. doi:10.1007/BF02291366.

Jöreskog KG (1973). "A General Method for Estimating a Linear Structural Equation System." In AS Goldburger, OD Duncan (eds.), *Structural Equation Models in the Social Sciences*, pp. 85–112. Seminar Press, New York.

Jöreskog KG (1993). "Testing Structural Equation Models." In KA Bollen, JS Long (eds.), *Testing Structural Equation Models*, pp. 294–316.

Jöreskog KG, Goldberger AS (1975). "Estimation of a Model with Multiple Indicators and Multiple Causes of a Single Latent Variable." *Journal of the American Statistical Association*, **70**(351), 631–639. doi:10.1080/01621459.1975.10482485.

Keesling JW (1972). *Maximum Likelihood Approaches to Causal Flow Analysis*. University of Chicago, Department of Education.

Knight K, Fu W (2000). "Asymptotics for Lasso-type Estimators." *The Annals of Statistics*, **28**(5), 1356–1378. doi:10.1214/aos/1015957397.

Konishi S, Kitagawa G (1996). "Generalised Information Criteria in Model Selection." *Biometrika*, **83**(4), 875–890. doi:10.1093/biomet/83.4.875.

Lee JD, Sun DL, Sun Y, Taylor JE (2016). "Exact Post-selection Inference, with Application to the Lasso." *The Annals of Statistics*, **44**(3), 907–927. doi:10.1214/15-AOS1371.

Leeb H, Pötscher BM (2006). "Can One Estimate the Conditional Distribution of Post-Model-Selection Estimators?" *The Annals of Statistics*, **34**(5), 2554–2591. doi:10.1214/009053606000000821.

Li L, Bentler PM (2006). "Robust Statistical Tests for Evaluating the Hypothesis of Close Fit of Misspecified Mean and Covariance Structural Models." In *UCLA Statistics Preprint #506*. Los Angeles: University of California.

Lin LC, Huang PH, Weng LJ (2017). "Selecting Path Models in SEM: A Comparison of Model Selection Criteria." *Structural Equation Modeling: A Multidisciplinary Journal*, **24**(6), 855–869. doi:10.1080/10705511.2017.1363652.

Magnus JR, Neudecker H (1999). *Matrix Differential Calculus with Applications in Statistics and Econometrics*. Second edition. John Wiley & Sons.

Mazumder R, Friedman JH, Hastie T (2011). "SparseNet: Coordinate Descent With Nonconvex Penalties." *Journal of the American Statistical Association*, **106**(495), 1125–1138. doi:10.1198/jasa.2011.tm09738.

McArdle JJ, McDonald RP (1984). "Some Algebraic Properties of the Reticular Action Model for Moment Structures." *British Journal of Mathematical and Statistical Psychology*, **37**(2), 234–251. doi:10.1111/j.2044-8317.1984.tb00802.x.

McDonald RP, Hartmann WM (1992). "A Procedure for Obtaining Initial Values of Parameters in the RAM Model." *Multivariate Behavioral Research*, **27**(1), 57–76. doi:10.1207/s15327906mbr2701_5.

Meng XL (1994). "On the Rate of Convergence of the ECM Algorithm." *The Annals of Statistics*, **22**(1), 326–339. doi:10.1214/aos/1176325371.

Meng XL (2008). "Discussion: One-step Sparse Estimates in Nonconcave Penalized Likelihood Models: Who Cares If It Is a White Cat or a Black Cat?" *The Annals of Statistics*, **36**(4), 1542–1552. doi:10.1214/07-AOS0316B.

Meng XL, Rubin DB (1993). "Maximum Likelihood Estimation via the ECM Algorithm: A General Framework." *Biometrika*, **80**(2), 267–278. doi:10.1093/biomet/80.2.267.

Meredith W (1993). "Measurement Invariance, Factor Analysis and Factorial Invariance." *Psychometrika*, **58**(4), 525–543. doi:10.1007/BF02294825.

Millsap RE (2011). *Statistical Approaches to Measurement Invariance*. New York: Routledge. doi:10.4324/9780203821961.

Muthén B, Asparouhov T (2012). "Bayesian Structural Equation Modeling: A More Flexible Representation of Substantive Theory." *Psychological Methods*, **17**(3), 313–335. doi:10.1037/a0026802.

Muthén L, Muthén B (1998-2010). *Mplus User's Guide. Eighth Edition*. Los Angeles, CA.

Neale MC, Hunter MD, Pritikin JN, Zahery M, Brick TR, Kirkpatrick RM, Estabrook R, Bates TC, Maes HH, Boker SM (2016). "**OpenMx** 2.0: Extended Structural Equation and Statistical Modeling." *Psychometrika*, **81**(2), 535–549. doi:10.1007/s11336-014-9435-8.

Negahban SN, Ravikumar P, Wainwright MJ, Yu B (2012). "A Unified Framework for High-Dimensional Analysis of M-Estimators with Decomposable Regularizers." *Statist. Sci.*, **27**(4), 538–557. doi:10.1214/12-STS400.

Neudecker H, Satorra A (1991). "Linear Structural Relations: Gradient and Hessian of the Fitting Function." *Statistics and Probability Letters*, **11**(1), 57–61. doi:10.1016/0167-7152(91)90178-T.

Pötscher BM (1991). "Effects of Model Selection on Inference." *Econometric Theory*, **7**(2), 163–185. doi:10.1017/S0266466600004382.

R Core Team (2017). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Rhemtulla M, Brosseau-Liard PÉ, Savalei V (2012). "When Can Categorical Variables Be Treated as Continuous? A Comparison of Robust Continuous and Categorical Sem Estimation Methods under Suboptimal Conditions." *Psychological Methods*, **17**(3), 354–373. doi:10.1037/a0029315.

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.

Rubin DB (1976). "Inference and Missing Data." *Biometrika*, **63**(3), 581. doi:10.2307/2335739.

Satorra A, Bentler PM (1994). "Corrections to Test Statistics and Standard Errors in Covariance Structure Analysis." In *Latent Variables Analysis: Applications to Developmental Research*, pp. 339–419.

Savalei V, Bentler PM (2009). "A Two-Stage Approach to Missing Data: Theory and Application to Auxiliary Variables." *Structural Equation Modeling: A Multidisciplinary Journal*, **16**(3), 477–497. doi:10.1080/10705510903008238.

Savalei V, Falk CF (2014). "Robust Two-Stage Approach Outperforms Robust Full Information Maximum Likelihood With Incomplete Nonnormal Data." *Structural Equation Modeling*, **21**(2), 280–302. doi:10.1080/10705511.2014.882692.

Schwarz G (1978). "Estimating the Dimension of a Model." *The Annals of Statistics*, **6**(2), 461–464. doi:10.1214/aos/1176344136.

Sclove SL (1987). "Application of Model-selection Criteria to Some Problems in Multivariate Analysis." *Psychometrika*, **52**(3), 333–343. doi:10.1007/BF02294360.

semTools Contributors (2016). ***semTools**: Useful Tools for Structural Equation Modeling*. URL https://cran.r-project.org/package=semTools.

Shapiro A, Browne MW (1983). "On the investigation of local identifiability: A counterexample." *Psychometrika*, **48**(2), 303–304. ISSN 1860-0980. doi:10.1007/BF02294025.

Sörbom D (1974). "A General Method for Studying Differences in Factor Means and Factor Structure Between Groups." *British Journal of Mathematical and Statistical Psychology*, **27**(2), 229–239. doi:10.1111/j.2044-8317.1974.tb00543.x.

Sörbom D (1989). "Model Modification." *Psychometrika*, **54**(3), 371–384. doi:10.1007/BF02294623.

Stone M (1977). "An Asymptotic Equivalence of Choice of Model by Cross-Validation and Akaike's Criterion." *Journal of the Royal Statistical Society B*, **39**(1), 44–47.

Strawderman RL, Wells MT, Schifano ED (2013). "Hierarchical Bayes, Maximum a Posteriori Estimators, and Minimax Concave Penalized Likelihood Estimation." *Electronic Journal of Statistics*, **7**, 973–990. doi:10.1214/13-EJS795.

Thurstone LL (1947). *Multiple-factor Analysis; A Development and Expansion of the Vectors of Mind*. University of Chicago Press, Chicago, IL, US.

Tibshirani R (1996). "Regression Selection and Shrinkage via the Lasso." *Journal of the Royal Statistical Society B*, **58**(1), 267–288. doi:10.1111/j.1467-9868.2011.00771.x.

Tutz G, Schauberger G (2015). "A Penalty Approach to Differential Item Functioning in Rasch Models." *Psychometrika*, **80**(1), 21–43. doi:10.1007/s11336-013-9377-6.

Varin C, Vidoni P (2005). "A Note on Composite Likelihood Inference and Model Selection." *Biometrika*, **92**(3), 519–528. doi:10.1093/biomet/92.3.519.

Vrieze SI (2012). "Model selection and psychological theory: A discussion of the differences between the Akaike Information Criterion (AIC) and the Bayesian Information Criterion (BIC)." *Psychological Methods*, **17**(2), 229–243. doi:10.1037/a0027127.

West SG, Taylor AB, Wu W (2012). "Model Fit and Model Selection in Structural Equation Modeling." In *Handbook of Structural Equation Modeling*, pp. 209–231. Guilford Press, New York, NY, US.

Wiley DE (1973). "The Identification Problem for Structural Equation Models with Unmeasured Variables." In A S Goldberger and O D Duncan (ed.), *Structural Equation Models in the Social Sciences*, pp. 69–83. Academic Press, New York.

Yuan GX, Ho CH, Lin CJ (2012). "An Improved **GLMNET** for L1-regularized Logistic Regression." *Journal of Machine Learning Research*, **13**(1), 1999–2030.

Yuan KH, Bentler PM (2000). "Three Likelihood-Based Methods for Mean and Covariance Structure Analysis with Nonnormal Missing Data." *Sociological Methodology*, **30**(1), 165–200. doi:10.1111/0081-1750.00078.

Yuan KH, Hayashi K (2006). "Standard Errors in Covariance Structure models: Asymptotics Versus Bootstrap." *British Journal of Mathematical and Statistical Psychology*, **59**(2), 397–417. doi:10.1348/000711005X85896.

Yuan KH, Hayashi K, Bentler PM (2007). "Normal Theory Likelihood Ratio Statistic for Mean and Covariance Structure Analysis under Alternative Hypotheses." *Journal of Multivariate Analysis*, **98**(6), 1262–1282. doi:10.1016/j.jmva.2006.08.005.

Yuan KH, Lu L (2008). "SEM with Missing Data and Unknown Population Distributions Using Two-stage Ml: Theory and Its Application." *Multivariate Behavioral Research*, **43**(4), 621–652. doi:10.1080/00273170802490699.

Yuan KH, Marshall LL, Bentler PM (2003). "Assessing the Effect of Model Misspecifications on Parameter Estimates in Structural Equation Models." *Sociological Methodology*, **33**(1), 241–265. doi:10.1111/j.0081-1750.2003.00132.x.

Zhang CH (2010). "Nearly Unbiased Variable Selection under Minimax Concave Penalty." *The Annals of Statistics*, **38**(2), 894–942. doi:10.1214/09-AOS729.

A. Standard errors and robust degrees of freedom

This appendix describes technical details of computing the standard errors and the so-called robust degrees of freedom in **lslx**. Let $\hat{\vartheta}$ denote a vector formed by the freely estimated and penalized non-zero elements of PL estimate $\hat{\theta}$. The expected Fisher information matrix under $\hat{\theta}$ is

$$\widehat{\mathcal{F}} = \left(\frac{\partial \tau(\hat{\theta})}{\partial \vartheta^\top}\right)^\top W(\hat{\theta}) \frac{\partial \tau(\hat{\theta})}{\partial \vartheta^\top}. \tag{25}$$

Similarly, the corresponding observed information is

$$\widehat{\mathcal{H}} = \frac{1}{2}\frac{\partial^2 \mathcal{D}(\hat{\theta})}{\partial \vartheta \partial \vartheta^\top}. \tag{26}$$

In **lslx**, $\widehat{\mathcal{F}}$ is computed via analytical formulas and $\widehat{\mathcal{H}}$ is obtained by numerical differentiation. By inverting $\widehat{\mathcal{F}}$ or $\widehat{\mathcal{H}}$, normal-theory standard errors can be obtained from the diagonal elements of the inverse matrix.

By default, **lslx** uses a sandwich covariance matrix to construct standard errors. Let $\hat{\tau}$ denote a consistent estimate of population moment vector τ such that $\sqrt{N}(\hat{\tau} - \tau) \to \mathcal{N}(0, \Pi)$. If $\hat{\tau}$ is calculated by the method described in Section 5.1, Π can be estimated by

$$\widehat{\Pi} = \left(\frac{\partial^2 \mathcal{L}(\hat{\tau})}{\partial \tau \partial \tau^\top}\right)^{-1} \left(\frac{1}{N}\sum_{n=1}^{N} \frac{\partial \mathcal{L}_n(\hat{\tau})}{\partial \tau}\frac{\partial \mathcal{L}_n(\hat{\tau})}{\partial \tau^\top}\right) \left(\frac{\partial^2 \mathcal{L}(\hat{\tau})}{\partial \tau \partial \tau^\top}\right)^{-1}, \tag{27}$$

where $\mathcal{L}_n(\tau) = -\frac{1}{2}\log|\Sigma_n(\tau)| - \frac{1}{2}[y_n^o - \mu_n(\tau)]^\top \Sigma_n(\tau)^{-1}[y_n^o - \mu_n(\tau)]$ (e.g., Yuan and Lu 2008). In **lslx**, the sandwich covariance matrix is obtained by

$$\hat{\mathcal{V}} = \widehat{\mathcal{H}}^{-1}\frac{\partial \tau(\hat{\theta})}{\partial \vartheta^\top} W(\hat{\theta}) \widehat{\Pi} W(\hat{\theta}) \frac{\partial \tau(\hat{\theta})}{\partial \vartheta} \widehat{\mathcal{H}}^{-1}. \tag{28}$$

Let $\hat{v}_{ii}$ denote the i^{th} diagonal element of $\hat{\mathcal{V}}$. $\sqrt{\hat{v}_{ii}/N}$ can be used as a standard error for $\hat{\vartheta}_i$, the i^{th} element of $\hat{\vartheta}$. Without the presence of penalization and model selection, the use of $\hat{\mathcal{V}}$ for two-stage estimation can be justified (Yuan and Bentler 2000).

The robust degrees of freedom is defined as the asymptotic expectation of LR statistics under null hypothesis. The expectation can be approximated by

$$df(\hat{\theta}) = \mathrm{tr}\left[\widehat{\Pi}\left(W(\hat{\theta}) - W(\hat{\theta})\frac{\partial \tau(\hat{\theta})}{\partial \vartheta}\widehat{\mathcal{F}}^{-1}\frac{\partial \tau(\hat{\theta})}{\partial \vartheta^\top}W(\hat{\theta})\right)\right], \tag{29}$$

(e.g., Yuan *et al.* 2007). This robust degrees of freedom (or equivalent) is often used for model selection with misspecified likelihood (e.g., Konishi and Kitagawa 1996; Stone 1977; Varin and Vidoni 2005).

B. Bounds for penalty and convexity level

This appendix describes how to obtain approximated values of λ_K and δ_1 for Λ and Δ initialization. Let β_{ij} denote the (i, j) element of B. According to the ECM algorithm for SEM

with PL (Huang *et al.* 2017), the thresholding rule for β_{ij} under MCP is

$$\hat{\beta}_{ij} = \begin{cases} \frac{\text{sign}(\tilde{\beta}_{ij})\max\{|\tilde{\beta}_{ij}| - \hat{w}_{\beta_{ij}}\lambda, 0\}}{1 - \tilde{w}_{\beta_{ij}}/\delta} & \tilde{\beta}_{ij} \leq \lambda\delta, \\ \tilde{\beta}_{ij} & \tilde{\beta}_{ij} > \lambda\delta, \end{cases} \tag{30}$$

where $\tilde{\beta}_{ij}$ is the current unpenalized estimate and $\hat{w}_{\beta_{ij}}$ is a working weight for β_{ij}. Under uncorrelated residuals, the working weight can be written as $\hat{w}_{\beta_{ij}} = \frac{\hat{\phi}_i^2}{c_j^2}$, where $\hat{\phi}_i^2$ is the current estimate for ϕ_i^2, and c_j^2 is the j^{th} diagonal element of $C = \mathbb{E}(\frac{1}{N}\sum_{n=1}^{N}\eta_n\eta_n^\top|\mathcal{Y}, \hat{\theta})$.

Let μ_i and σ_i denote the mean and the standard deviation of η_i, the i^{th} element of η. The standardized $\tilde{\beta}_{ij}$ can be written as $\tilde{\beta}_{ij}^* = \frac{\sigma_j}{\sigma_i}\tilde{\beta}_{ij}$. If we hope to shrink all $|\tilde{\beta}_{ij}^*| \leq t$ to be zero, Equation 30 indicates λ_K should at least satisfy

$$\lambda_K \geq \max_{i,j} \frac{c_j^2\sigma_i}{\hat{\phi}_i^2\sigma_j}t \approx \max_{i,j} \frac{\sigma_j + \mu_j^2/\sigma_j}{\sigma_i(1 - r_i^2)}t. \tag{31}$$

where r_i^2 is the coefficient of determination for η_i. The approximation is based on $c_j^2 \approx \sigma_j^2 + \mu_j^2$. For a system such that all exogenous variables are centered, a loose approximation can be obtained by $\lambda_K \approx \frac{\sigma_{\max}}{\sigma_{\min}(1 - r_{\max}^2)}t$.

By Equation 30, $1 - \hat{w}_{\beta_{ij}}/\delta$ must be larger than zero. Therefore, δ_1 can be approximated by

$$\delta_1 \geq \max_{i,j} \frac{\hat{\phi}_i^2}{c_j^2} \approx \max_{i,j} \frac{\sigma_i^2(1 - r_i^2)}{\sigma_j^2 + \mu_j^2}. \tag{32}$$

Again, without the consideration of μ_j, we can loosely use $\delta_1 \approx \frac{\sigma_{\max}^2(1 - r_{\min}^2)}{\sigma_{\min}^2}$.

In principle, **lslx** initializes Λ and Δ based on the approximations in Equations 31 and 32. When variable scales are all the same, these approximations become very simple. Hence, standardization is a good strategy to simplify the initialization problem. Note that the approximations can be further improved if exogenous and endogenous variables are distinguished. However, how to obtain good estimates for σ_i^2 and r_i^2 without actual model fitting is still a challenging task.

Affiliation:

Po-Hsien Huang
Department of Psychology
National Cheng Kung University
No.1, University Road, Tainan City 701, Taiwan
E-mail: psyphh@gmail.com