NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

LINEAR NON-GAUSSIAN ACYCLIC MODELS FOR CAUSAL INFERENCE OF LATENT VARIABLES IN STRUCTURAL EQUATION MODEL

LUK CHUN TO

SCHOOL OF SOCIAL SCIENCES

2023

LINEAR NON-GAUSSIAN ACYCLIC MODELS FOR CAUSAL INFERENCE OF LATENT VARIABLES IN STRUCTURAL EQUATION MODEL

Luk Chun To

SCHOOL OF SOCIAL SCIENCES

A thesis submitted to the Nanyang Technological University in partial fulfilment of the requirement for the degree of Master of Arts

2023

Statement of Originality

I certify that all work submitted for this thesis is my original work. I declare that no other person's work has been used without due acknowledgement. Except where it is clearly stated that I have used some of this material elsewhere, this work has not been presented by me for assessment in any other institution or University. I certify that the data collected for this project are authentic and the investigations were conducted in accordance with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

26 April 2023

. . . . . .
Date

. . . . . .
Luk Chun To

Supervisor Declaration Statement

I have reviewed the content of this thesis and to the best of my knowledge, it does not contain plagiarised materials. The presentation style is also consistent with what is expected of the degree awarded. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accordance with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

26 April 2023

. . . . . .
Date

. . . . . .
Prof. Moon-ho Ringo HO

Authorship Attribution Statement

This thesis **does not** contain any materials from papers published in peer-reviewed journals or from papers accepted at conferences in which I am listed as an author.

26 April 2023

. . . . . .
Date

. . . . . .
LUK Chun To

Acknowledgements

I would like to acknowledge and express my gratitude to my supervisor Prof Ringo for his guidance, advice and timely feedback, despite his extremely packed working schedule. His advice and suggestive readings related to machine learning implicitly but truthfully contribute the idea of the thesis topic.

I deeply appreciate the support from my mum and dad, socially and emotionally. Without them, I will have never overcome challenges after challenges during the two-year study.

Last but not least, I would also like to give thanks to NTU research scholarship for funding this research.

Table of Contents

Table of Contents 5

List of Tables 8

List of Figures 9

Abstract 11

Chapter 1: Literature Review of Structural Equation Models 13

1.1 Motivation 13

1.2 Study Objectives 17

1.3 Introduction to Structural Equation Modelling 19

1.4 Graphical Model 20

1.5 Causal Inference in Structural Equation Modelling 24

1.6 Misspecifications in Structural Equation Model 26

1.7 Ways to Identify Path-Model Misspecifications 27

Chapter 2: Literature Review of Existing Causal Discovery Methods 34

2.1 Gaussian Causal Discovery 34

2.2 Non-Gaussian Causal Discovery 38

Chapter 3: Present Study 57

3.1 Adapting LiNGAM to Latent Variable Models 57

3.2 ICA-LiNGAM for Latent Variables 58

3.3 DirectLiNGAM for Latent Variables 65

3.4 ParceLiNGAM for Latent Variables 72

Chapter 4: Method 77

4.1 Study Design 77

4.2 Data Generation 81

4.3 Data Analysis 82

4.4 Performance Criteria 83

Chapter 5: Results 85

5.1 Preliminary Analysis 85

5.2 Population Model 1 86

5.3 Population Model 2 92

5.4 Population Model 3 98

Chapter 6: Discussion 103

6.1. Study Overview 103

6.2 Summary of Study Findings 104

6.3 Contributions, Implication and Recommendation 108

6.4 Limitation and Future Direction 111

6.5 Conclusion 114

References 115

Appendices 131

Appendix 1: Operational Definition of Structural Coefficients 131

Appendix 2: Sequence of Nested Models 132

Appendix 3: Solutions to Permutation and Scaling Indeterminacies 137

Appendix 4: Approximation for Matrix Permutation to Strict Lower Triangularity 138

Appendix 5: Prove of Mutual Information as A Function of One-dimensional Entropies 140

Appendix 6: Algorithm of ICA-LiNGAM 142

Appendix 7: Algorithm of DirectLiNGAM 144

Appendix 8: Algorithm of ParceLiNGAM 146

Appendix 9: Algorithm of ICA-LiNGAM-LV 148

Appendix 10: Algorithm of DirectLiNGAM-LV.....150

Appendix 11: Algorithm of ParceLiNGAM-LV.....152

Appendix 12: R Documentation of LiNGAM Algorithms 154

Appendix 13: Mean and Standard Deviation of Performance Indicators 159

Appendix 14: Three Most Frequently Discovered Path Models by Sample Size.....171

Appendix 15: LiNGAM Algorithms with R in Action 184

Appendix 16: R code for LiNGAM algorithms 194

List of Tables

Table 1 Algorithm Overview of ICA-LiNGAM 47

Table 2 Algorithm Overview of DirectLiNGAM 52

Table 3 Algorithm Overview of ParceLiNGAM 56

Table 4 Algorithm Overview of ICA-LiNGAM-LV 65

Table 5 Algorithm Overview of DirectLiNGAM-LV 72

Table 6 Algorithm Overview of ParceLiNGAM-LV 76

Table 7 Multivariate Skewness and Multivariate Kurtosis 80

Table 8 Hyperparameters Specified in the Simulation Study 83

Table 9 Confusion Matrix for Binary Classifier on a Pathway 84

List of Figures

Figure 1 Edges in Graphical Models....21

Figure 2 A Directed Acyclic Graph to Illustrate d-separation....22

Figure 3 Schematic Representation of A Mediation Model....24

Figure 4 Structural Equation Modelling Inference Process....25

Figure 5 Illustration of Causal Discovery with PC Algorithm....35

Figure 6 Illustration of Causal Discovery with FCI Algorithm....36

Figure 7 Illustration of Causal Discovery with LiNGAM Algorithm....40

Figure 8 Path Diagram of Population Model 1....78

Figure 9 Path Diagram of Population Model 2....78

Figure 10 Path Diagram of Population Model 3....79

Figure 11 Global RMSEA of Measurement Models over 500 Replications....85

Figure 12 Global CFI of Measurement Models over 500 Replications....86

Figure 13 RMSEA-P on Population Model 1....87

Figure 14 CFI-P on Population Model 1....88

Figure 15 Accuracy on Population Model 1....88

Figure 16 Root Mean Square Error on Population Model 1....89

Figure 17 Three Most Frequently Discovered Path Models on Population Model 1....91

Figure 18 RMSEA-P on Population Model 2....93

Figure 19 CFI-P on Population Model 2....94

Figure 20 Accuracy on Population Model 2....95

Figure 21 RMSE on Population Model 2....96

Figure 22 Three Most Frequently Discovered Path Models on Population Model 2....97

Figure 23 RMSEA-P on Population Model 3....98

Figure 24 CFI-P on Population Model 3....99

Figure 25 Accuracy on Population Model 3 100

Figure 26 Root Mean Square Error on Population Model 3 101

Figure 27 Three Most Frequently Discovered Path Models on Population Model 3 102

Abstract

In psychology and social sciences, confirmatory data analysis and hypothesis testing are in active use, but sometimes prior studies are not available under which researchers may consider exploratory approach to analysing the data. Existing causal discovery methods designed to explore directional relationships between variables are capable of handling mainly observed variables but not latent. Latent variables are equally prevalent as some psychological constructs are not directly observable. Some current methods to determine the cause-effect sequences of latent variables (e.g., direction dependence analysis and structural equation modelling with higher-order moment structures) have been under-utilized due to the large sample size requirement and lack of statistical software support. To close the gap in the literature of causal discovery and of structural equation modelling with latent variables, this thesis develops a data-driven latent-variable causal discovery method – linear non-Gaussian acyclic models for latent variables (LiNGAM-LV). With the input of raw data and user-specified measurement model, LiNGAM-LV algorithms output the path models with latent variables in the forms of directed acyclic graph based on the criteria of pathway directionality and the balance between model complexity and model fit. This thesis proposes three types of LiNGAM-LV algorithms (i.e., ICA-LiNGAM-LV, DirectLiNGAM-LV and ParceLiNGAM-LV) and provides R codes for the implementation. A Monte Carlo simulation study follows that varies sample sizes, non-normality, data missingness and model complexity. It is aimed at evaluating the performance of the three algorithms with respect to path-related fit indices, accuracy and root mean square error. The simulation results revealed that the LiNGAM-LV algorithms gave a promising performance in general; the performance of DirectLiNGAM-LV and ICA-LiNGAM-LV were comparable while ParceLiNGAM-LV was the worst. LiNGAM-LV algorithms provide insight into the causal relationships between latent variables without

relying on priori hypotheses. Recommendations on using LiNGAM-LV in practice are discussed.

Keywords: causal discovery, latent variable, non-Gaussianity, directional dependence, structural equation modelling, exploratory.

Chapter 1: Literature Review of Structural Equation Models

1.1 Motivation

Usage of latent variables has been proliferating in sciences, including psychology. Generally speaking, latent variables are some hypothetical constructs that one cannot directly measure or observe (Bollen, 2002). One way to model latent variables is through covariance-based structural equation modelling (SEM). A latent-variable model usually consists of two portions: i) a measurement model that addresses the relations between observed and latent variables, and ii) a path model that primarily address the directional relationships between latent variables. The path model often bears the responsibility of hypothesis testing in confirmatory data analysis. Specifying a theoretically plausible path model for hypothesis testing is imperative as the (dis)confirmation of the proposed hypotheses yield support to the theories behind. Some non-statistical deductive methods such as elimination approach (Henley et al., 2006) guide researchers to reason and justify the effects between variables and the directions based on conceptual theories, previous studies and prior knowledge,

On some occasions, prior studies are not available. With no hypotheses being made, the effects between latent variables and their direction in the path model can be ambiguous. Between two variables, say conflict and negative emotion, it is possible that, conflict is not related to negative emotion (less likely), conflict brings one negative emotion, or negative emotion incites conflict. Certainly, one can test the effect and the direction with longitudinal data. But what happens if, like most of the data collection, the study variables are measured once only? Can model fit evaluation in SEM help with the problem? We will briefly go through some issues of model fit evaluation in SEM.

Issue of Misspecification with Model Fit. In SEM, model fit indicates how well the research model that one fits describes the given data. Yet, model fit cannot locate the source of misspecification in the research model, if any (Mulaik et al., 1989). As a latent-variable

model consists of a measurement model portion and a path model portion, if the fit is poor, we have little idea as to which portion of the latent-variable research model is wrongly specified and deserves attention to fix. What make things worse, if the directional relationship between latent variables in the path model is falsely hypothesized, as far as the measurement model portion is consistent with the given data, it can still yield an acceptable fit. In contrast, if the causal relationship is perfectly correct, as far as the measurement model portion is *not* consistent with the data, one can receive poor fit. The phenomenon lies on the fact that traditional fit indices are not quite sensitive to the misspecification happening in the path model, largely due to the disproportionately low number of model parameters in the path model in comparison to that of the measurement model (e.g., McDonald & Ho, 2002).

Issue of Direction Dependence with Model Fit. Assigning directions to the effects between variables, or the pathways in SEM, touches on an issue of direction dependence. Direction dependence is a term that describes the cause-effect sequence between two variables, as to which is the cause and which is the effect (von Eye & DeShon, 2012). Nonetheless, assigning a wrong direction sometimes does not warrant a change in model fit. In what follows, we will illustrate a phenomenon that a pathway in wrong direction sometimes cannot be captured by the model fit nor the standardized parameter estimates.

Suppose we have two observed variables, say x_i and x_j. x_i is the true cause and x_j is the true effect. Without loss of generality, we set their means to zero. Surrounding the two variables, one can construct two models (M[$xi \rightarrow xj$] & M[$xj \rightarrow xi$]), one contradictory to the other.

$$\mathrm{M}[xi \rightarrow xj]: x_j = \gamma_{ji} x_i + e_j^{(i)}$$

$$\mathrm{M}[xj \rightarrow xi]: x_i = \gamma_{ij} x_j + e_i^{(j)}$$

where $\gamma_{ji} = \frac{\mathrm{cov}(x_j,x_i)}{\sigma_i^2}$ and $\gamma_{ij} = \frac{\mathrm{cov}(x_i,x_j)}{\sigma_j^2}$ denotes the regression coefficients, and *e* denotes the residual. If the variables are standardized (i.e., $\sqrt{\sigma_i^2} = \sqrt{\sigma_j^2} = 1$), the point estimate of γ_{ji} and of γ_{ij} are equal, so are their *t* statistics (von Eye & De Shon, 2012). Regression itself does not suffice to decide the causal direction between x_i and x_j.

The same ambiguity also applies to latent variables. Between two latent variables (η_i & η_j), suppose η_j is the true cause. One can build two contradictory models, M[$\eta i \rightarrow \eta j$] and M[$\eta j \rightarrow \eta i$].

$$\mathrm{M}[\not\eta i \rightarrow \not\eta j]: \eta_j = \beta_{ji}\eta_i + d_j^{(i)}$$

$$\mathrm{M}[\not\eta j \rightarrow \not\eta i]: \eta_i = \beta_{ij}\eta_j + d_i^{(j)}$$

where β denotes the structural parameter and *d* is the disturbance term. Without loss of generality, the two latent variables have zero means. Since the variance-covariance matrices in M[$\eta i \rightarrow \eta j$] is identical to that of M[$\eta j \rightarrow \eta i$], the standardized point estimate of β_{ji} and β_{ij} are the same, as are the fit statistics of the two models. In fact, M[$\eta i \rightarrow \eta j$] and M[$\eta j \rightarrow \eta i$] qualify for equivalent models. Equivalent models refer to a pair of or multiple models that reproduce and are compatible with the same set of variance-covariance matrices (Hershberger, 2006; Raykov & Marcoulides, 2001; Stelzl, 1986; see also Asparouhov & Muthén, 2019; Bentler & Satorra, 2010). In sum, the fit statistics and the standardized parameter estimates are of no use in differentiating between the two models, nor telling which one is preferred. Judea Pearl (2014) echoes the captioned idea that acceptable fit indicative of good match between data and model does *not* sufficiently prove the robustness of the tested model nor the validity of structural parameters and their direction dependence.

Issue of Alternative Models with Model Fit. Obtaining a good model fit does not mean that the research model is the only one that fits the data well. Relevant to the issue with direction dependence, when a path model involves more than two variables, there are multiple alternative models that could give a model fit as good as the research model (Pearl, 2014). Alternative models refer to those other than the research model that describe the given data equally well or even better. Without testing alternative models, it is questionable to place much confidence on the research model despite with good fit. Some reasons as to why alternative models are left unexplored are the lack of theoretical foundation and the soaring number of alternative models that grows with the increasing number of variables. Testing many alternative models manually could be time inefficient. Of course, it is also possible that alternative models are being tested yet not reported due to the space limit. In short, these factors potentially render alternative models unexplored, and the gathered data will not be used to its fullest potential.

More Methods for Direction Dependence. Given the three issues surrounding the model fit evaluation in SEM, one cannot solely rely on model fit to find out the presence/absence of the effect between variables and, if present, the cause-effect sequences. At the time of writing, a few substitution can be found in the literature for identification of direction dependence. Statistical methods take advantages of the mathematical characteristics of latent variables to judge the direction, such as *direction dependence analysis* (Wiedermann & von Eye, 2015), and *SEM with higher-order moment structures* (Shimizu & Kano, 2008). Yet, these statistical methods have their own limitation. Direction dependence analysis can only examine directionality in pairwise fashion (i.e., two variables at a time). SEM with higher-order moment structures requires sufficiently large sample size to operate, e.g., about 5,000 observations in a three-factor model with 15 observed variables (Yuan & Bentler, 1998).

1.2 Study Objectives

Increasingly, it is the responsibility of applied researchers to justify the directional relationship using relevant literatures and prior studies (Henley et al., 2006). Sometimes, there are insufficient or no prior studies to inform of the presence/absence of effects and the directions; sometimes prior studies can give contradictory prediction of the effect. As discussed earlier, model fit in SEM possesses some ability to detect the presence of pathways in the path model but it is not a preferred option to determine the direction dependence. Some current methods for direction dependence have limitation on a practical level. Most of the the causal discovery methods to be reviewed later (e.g., PC algorithm & LiNGAM algorithms) that aim to discover the directional relationship between variables are accommodative of observed variables but not latent variables. To the best of our knowledge, there is no statistical tool available to detect the presence of effects between latent variables and at the same time assign directions to the non-zero effects, without resorting to previous studies and relevant substantive theories.

To fill in the gap in the literature of SEM and of causal discovery, this thesis draws as inspiration the linear non-Gaussian acyclic model (LiNGAM; Shimizu et al., 2006) and develops a causal discovery approach for latent variables by adapting the existing LiNGAM to the context of latent-variable models. The newly proposed method is named LiNGAM for latent variables (LiNGAM-LV). LiNGAM-LV is intent on discovering a path model with latent variables conditional on the user-specified measurement model that LiNGAM-LV fits the given raw data the most in terms of two criteria. The first criterion is the accuracy of the pathway directionality as derived from the data property of multivariate non-Gaussian distribution. The second criterion is the balance between model complexity and model fit based on regularization. The objective of this thesis is threefold: i) to derive three new LiNGAM-LV algorithms for the latent-variable causal discovery, namely ICA-LiNGAM-LV

(Shimizu et al., 2006), DirectLiNGAM-LV (Shimizu et al., 2011) and ParceLiNGAM-LV (Tashiro et al., 2014); ii) to develop R codes to implement the aforementioned algorithms, and iii) to conduct a simulation study to evaluate and compare the performance of the three algorithms.

Through the development of a new causal discovery approach for latent-variable models, we contribute the literature of SEM and the model fit evaluation in two ways. First, it is an exploratory statistical tool that does not require prior hypotheses concerning the path model. With the raw data plus the measurement model one specifies, LiNGAM-LV algorithms discover path models that offer one insight into what the pattern of latent variables and their causal relationships would look like from the data perspective. Sometimes, there is insufficient prior knowledge to guide researchers to come up with a research model. When the research model is absent, the proposed algorithm aids researchers in exploring potential patterns, as a starting point of data analysis.

Secondly, the path model discovered by the LiNGAM-LV algorithms can be thought of as an alternative model to the research path model, if any. The acknowledgement of alternative models aids researchers to improve their study design and/or set up follow-up studies to rule out alternative models, thereby consolidating the theoretical foundation (Henley et al., 2006). Otherwise, or if one chooses to ignore the existence of alternative models, their results findings could be questionable.

We will review the literature of SEM and graphical models in the remainder of Chapter 1, review the literature of causal discovery methods in Chapter 2, develop LiNGAM-LV and its algorithms in Chapter 3, conduct and report a simulation study in Chapters 4 and 5, and discuss the algoritms in Chapter 6.

1.3 Introduction to Structural Equation Modelling

Structural equation model refers to a general latent-variable model in which statistical assumptions (Bentler & Chou, 1987) include linearity and independent and identically distributed ("i.i.d.") observations. LISREL model (Jöreskog, 1973) or the Jöreskog-Keesling-Wiley model expresses a structural model into two portions: i) a measurement model and ii) a path model. In what follows, we use the mathematical notations that letters in lowercase bold (e.g., **x**) denote vectors and letters in uppercase bold (e.g., **X**) denote matrices. A measurement model captures the relationships between observed and latent variables. Suppose $\mathbf{x} = (x_1, \ldots, x_p)^{\mathrm{T}}$ is a $(p \times 1)$ random (or unknown) vector of continuous observed variables, where "T" is a transpose operator. Here, observed variables are presumably mean-centred without loss of generality. A measurement model can be expressed in equation form as:

$$\mathbf{x} = \boldsymbol{\Lambda}\boldsymbol{\eta} + \boldsymbol{\varepsilon} \tag{1.1}$$

where $\boldsymbol{\eta} = (\eta_1, \ldots, \eta_p)^{\mathrm{T}}$ is a $(q \times 1)$ random vector of continuous latent variables with zero means. Even though the LISREL model distinguishes between endogenous and exogenous latent variables, the focus of this thesis is on causal discovery and thus no prior knowledge guides one to determine the nature of latent variables as endogenous or exogenous. Thereby, $\boldsymbol{\eta}$ is inclusive of endogenous and exogenous latent variables. $\boldsymbol{\varepsilon}$ is a $(p \times 1)$ random vector of measurement errors, which has zero mean and is uncorrelated with $\boldsymbol{\eta}$ (i.e., $\mathrm{cov}(\boldsymbol{\varepsilon}, \boldsymbol{\eta}) = \mathbf{0}$) and is in normal distribution (Bollen, 1989). $\boldsymbol{\Lambda}$ is a $(p \times q)$ factor loading matrix matrix that portrays linear relations between observed and latent variables. A path model primarily addresses directional relationships between latent variables which can be expressed in equation:

$$\boldsymbol{\eta} = \mathbf{B}\boldsymbol{\eta} + \boldsymbol{\zeta} \tag{1.2}$$

where $\boldsymbol{\zeta}$ is a ($q \times 1$) random vector of disturbances on latent variables and is assumed not correlated with $\boldsymbol{\eta}$, with zero means and in normal distribution. $\mathbf{B}$ is a ($q \times q$) matrix of structural parameters. of which diagonal elements are restricted to zero, for it never allows a latent variable to exert a direct effect on itself.

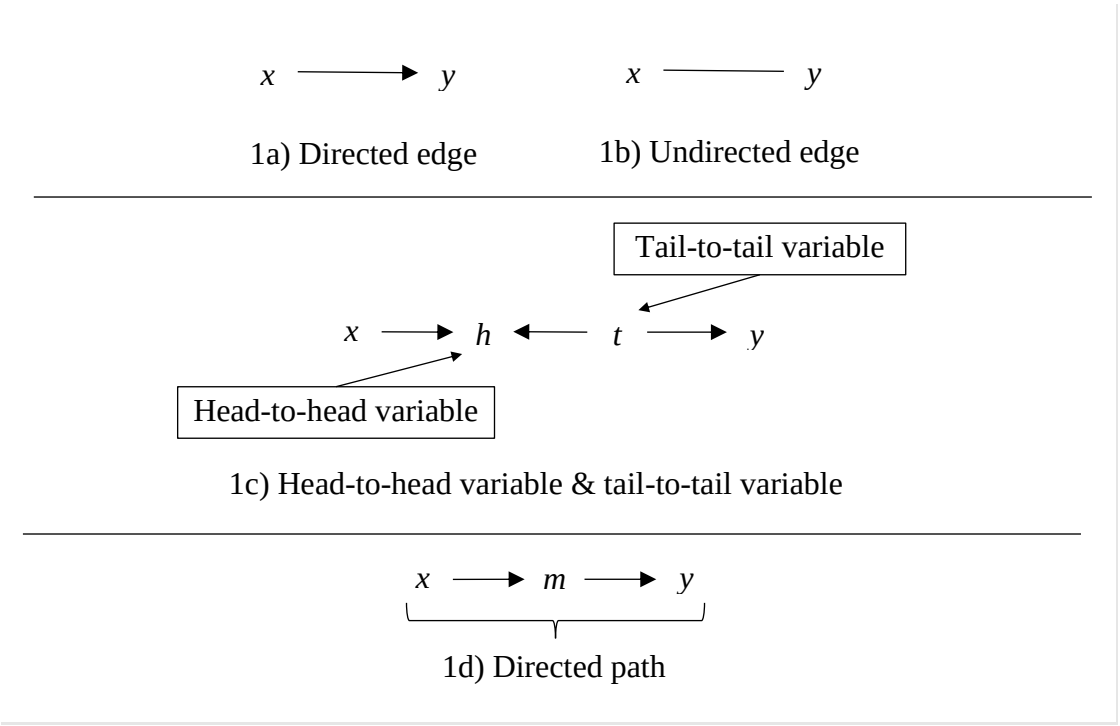
1.4 Graphical Model

1.4.1 Basic Components of Graphical Models

Using graphical models to picture the diagrams of SEM has been gaining popularity with psychology (Borsboom & Cramer, 2013). Graphical model is a probability model that uses a diagram to delineate the conditional dependence of variables and their causal relations (Greenland et al., 1999). It is composed of two fundamental items: nodes denoting variables and edges denoting connections between variables. A directed edge in graphical model is a pathway connecting two variables that *x* has a direct effect on *y* (see Figure 1a). An undirected edge is a non-directional association between two variables which symbolizes the correlation between *x* and *y* (see Figure 1b). A head-to-head variable is defined as a variable receiving two (or more) arrowheads, like *h* in Figure 1c. A tail-to-tail variable is defined as the one extending two (or more) directed edges, like *t* in Figure 1c. A directed path in graphical models refers to a sequence of directed edges formed between variables which starts at *x*, via *m* and ends at *y* (see Figure 1d). One is reminded not to confuse the term "directed path" used in graphical model with "pathway" in SEM. Technically speaking, a pathway is equivalent to a directed edge. Multiple pathways that do not form loops are a directed path. To circumvent confusion in the remaining of this thesis, we will adhere to the term "pathway" for the direct effect and will avoid the term "directed path". Interestingly, a biological concept about kinships is often applied to describe relations between variables. In Figure 1a, *x* is labelled as the *parent* of *y*, and *y* is the *child* of *x*. In Figure 1d, *x* and *m* are

said to be the ancestors of *y*; *m* and *y* are the descendants of *x* (Greenland et al., 1999; Lauritzen, 1996; Pearl, 1995).

Figure 1 *Edges in Graphical Models*



1a) Directed edge 1b) Undirected edge

1c) Head-to-head variable & tail-to-tail variable

1d) Directed path

The presentation of graphical models can be categorized into two types: *directed acyclic graph* and *undirected graph*. By their names, directed acyclic graph (DAG) is the one that expresses relations between variables using directed edges only. An undirected graph is the one that describes relations between variables with non-directional edges only. In psychology, an undirected graph serves to formalize network structures in Gaussian graphical model in an emerging field called network psychometric (Epskamp et al., 2017). Yet, undirected graph is out of the scope, and we will be focusing on DAG only.

1.4.2 Directed Acyclic Graph

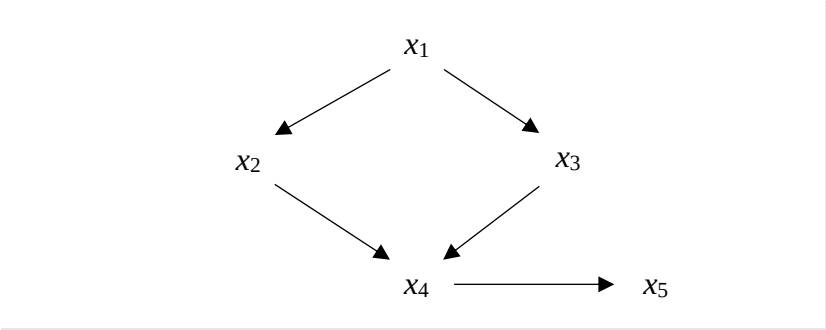
Suppose $\mathbf{x} = (x_1,\ldots,x_p)^{\mathrm{T}}$ is a ($p \times 1$) random vector of observed variable. Let P be the joint probability distribution for **x**. DAG (or the Spirtes-Glymour-Scheines model; Robins et al., 2003) is a representation of the conditional independences in **x** in forms of recursive product decomposition (Pearl, 1995):

$$P = \text{pr}(x_1, \ldots, x_p) = \prod_{i}^{p} \text{pr}(x_i | \text{parent}(x_i)) \quad (1.3)$$

where "pr" denotes the probability density function and "parent (x_i)" denotes a set of parents of x_i in the DAG. Should x_i has no parent, pr(x_i | parent(x_i)) is simplified to pr(x_i). In words, *P* is a function of the product of conditional probability of variables in **x** given their parents. Since x_i cannot have children that are the parents of x_i itself, the term pr(x_i | parent(x_i)) in equation 1.14 captures a property of DAG known as recursiveness (Bollen, 1989) or acyclicity (Lauritzen, 1996), by which directed edges cannot form loops (see VanderWeele & Robins, 2007).

Rule of *d*-separation. DAG complies with the rule of *d*-separation, a vastly adopted definition in graphical models The compliance with the rule of *d*-separation is to guarantee that the DAG is a portrait of causal relationships (Geiger et al., 1990; Pearl, 2000). Briefly, a set of variables as X is said to be *d*-separated from the other set of variables as Y by the third set of variables as Z under one of the two circumstances: i) when a sequence of directed edges formed between variables contains tail-to-tail variables and/or ii) when a sequence of directed edges has head-to-head variables who are not elements of Z, nor have descendants inside of Z.

Figure 2 *A Directed Acyclic Graph to Illustrate d-separation*



In Figure 2 that shows a sequence of directed edges with x_2, x_4 and x_3, x_4 is the respective head-to-head variable because of the two consecutive links ($x_2 \rightarrow x_4$ & $x_4 \leftarrow x_3$). In the same

figure which has another sequence of directed paths with x_2, x_1 and x_3, x_1 is the tail-to-tail variable due to the two consecutive links ($x_2 \leftarrow x_1$ & $x_1 \rightarrow x_3$) on the trail. Suppose X = $\{x_2\}$, Y = $\{x_3\}$ and Z = $\{x_1\}$. Z *d*-separates X from Y as the path $x_2 \leftarrow x_1 \rightarrow x_3$ is blocked by $x_1 \in$ Z. Z *d*-separates X from Y in another way that the path $x_2 \rightarrow x_4 \leftarrow x_3$ is blocked by x_4 provided that neither x_4 nor its descendant, x_5, is inside Z. Suppose Z' = $\{x_1, x_5\}$, we can state that X and Y are not *d*-separated by Z' but are connected conditional on Z' since the head-to-head variable (i.e., x_4) has a descendant, x_5, inside Z'.

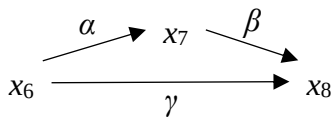
Causal Marko Condition. DAG follows two key assumptions: causal Marko condition (CMC) and causal faithfulness condition (CFC) (Pearl, 2000; Spirtes et al., 2000). CMC governs the probabilistic conditional independence between variables (Weinberger, 2018). It explicates that, if two sets of variables, say X and Y, are *d*-separated by a third set of variables, Z, in a DAG, then X and Y are independent conditional on Z. When X and Y are *d*-separated by parents of X, their conditional independence can be expressed as:

$$X \perp\!\!\!\perp Y \mid \text{parent}(X)$$

If conditional independence holds for every variable pair, CMC establishes in the DAG and *P* is said to be *Markov* to the DAG. (Robins et al., 2003). It is necessary to assume CMC in DAG for causal inference.

Causal Faithfulness Condition. CFC, also known as stability (Pearl, 2000) and faithfulness (Spirtes et al., 2000), governs which variables are probabilistically dependent of others conditional on a set of variables (Weinberger, 2018). Given a DAG in which variables are *d*-separated and thus conditionally independent based on CMC, CFC makes inference one step further by presuming that all other variables not being *d*-separated are dependent. However, CFC does not always hold. Of many counterexamples against CFC, Weinberger (2018) raises a classic epitome about cancellation.

Figure 3 *Schematic Representation of A Mediation Model*



Note: α = a regression parameter describing the effect of variable x_6 on x_7; β = a regression parameter describing the effect of x_7 on x_8; γ = a regression parameter describing the (direct) effect of x_6 on x_8. The multiplication of α by β is the indirect effect of x_6 on x_8 which is mediated by x_7.

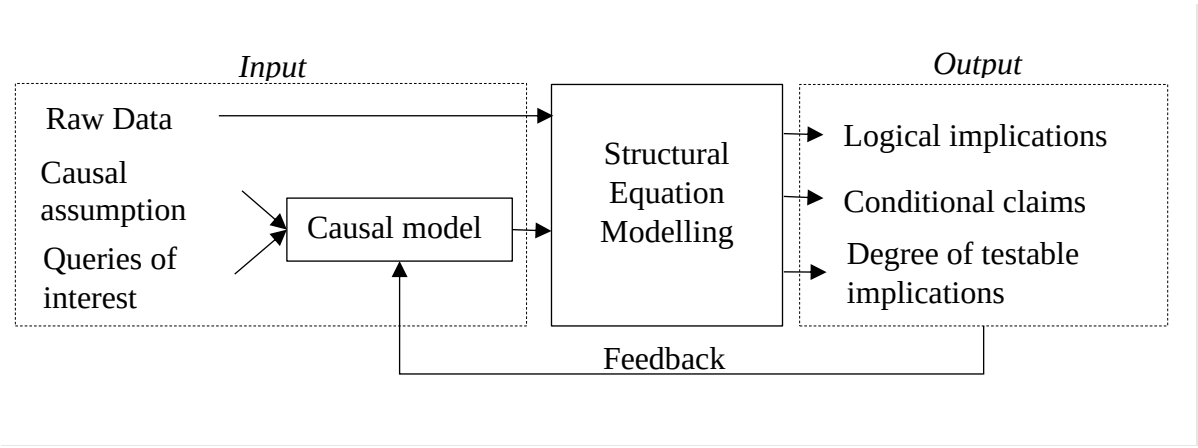
Figure 3 shows a mediation model with a predicting variable x_6, a mediating variable x_7, and an outcome variable x_8. If the magnitude of the indirect effect negatively equates to that of the direct effect, that is, $\alpha\beta = -\gamma$, x_6 and x_8 will be independent despite the causal link from x_6 to x_8. Because of that, DAGs usually do not demand CFC as an assumption, a circumstance known as a lack of faithfulness. Still, without assuming CFC, a DAG remains a representation of causal hypotheses surrounding variables which resembles a path model in SEM.

1.5 Causal Inference in Structural Equation Modelling

SEM is an inference method that draws from three types of input (causal assumptions, queries of interest & raw data) and produces three types of output (logical implications of causal assumptions, claims & testable implications) (Pearl, 2014). Two types of input, the causal assumptions and the queries of interest constitutes the diagram of a causal model for hypothesis testing which can be displayed in the form of DAG. Figure 4 displays the operational flow of SEM where it processes inputs of causal assumptions, queries of interest and raw data. The integrity of output resides on the input (Pearl, 2000). Importantly, it is a misconception that SEM establishes causal relations based solely on the magnitude of effects between variables (Bollen & Pearl, 2013, p. 309). Instead, SEM infers causality when and

only when accompanied with causal assumptions behind the proposed model. In the following are descriptions of the input and output of SEM.

Figure 4 *Structural Equation Modelling Inference Process*



1.5.1 Input of SEM

Causal Assumptions. Causal assumptions are a set of statements proposed and defended in hypotheses development which are usually derived from scientific justification, research paradigms, previous studies, prior knowledge, logics and alike (Bollen & Pearl, 2013). Precisely, one can categorize causal assumptions into two types. One is the strong causal assumption under which the estimate of a parameter is fixed to a specific value, say zero. The other is the weak causal assumption under which the estimate of a parameter wanders within a range of values (e.g., above or below zero; see Bollen & Pearl, 2013).

Queries of Interest. Queries of interest concerns causal and counterfactual relationships between variables that arouse researchers' curiosities (Pearl, 2014). Queries are usually about the direction of pathways (e.g., "Does variable *X* cause *Y*?") and the strength (e.g., "What is the effect of *X* on *Y*?").

Raw Data. Raw data can be gathered from either observational, quasi-experimental or purely experimental sources. A decent match between the causal model and the raw data implies that joint probability distribution of variables in the data conforms to the causal assumptions.

1.5.2 Output

Logical Implications. Logical implications are statements and conclusions of the causal model estimation given that the causal assumptions hold (e.g., "variable *X* causes *Y*"). Logical implications are not only applicable to the given data but also are generalizable to the concerned populations and extendable to the theoretical and practical implications.

Conditional Claims. Conditional claims are numerical outputs of fitting causal model into the raw data which give answers to queries of interest. Claims are made conditional on the causal assumptions. Some examples of conditional claims are the parameter estimate of the effect of *X* on *Y*, and the mean and variance of variables.

Testable Implications. Testable implications is the extent to which the raw data agrees with the causal assumptions. Model fit quantifies the degree to which the causal model matches the raw data. It is used to feedback the inference in SEM. If the model fit is good, the corresponding causal assumptions is likely reasonable (Bollen & Pearl, 2013). In contrast, if the model fit is poor, the causal assumptions are questionable. Obtaining a good model fit from SEM is not meant to validate the causal assumptions but to give credibility to the causal assumptions. If one repeatedly fit the same model to different datasets and still obtains good fit, the credibility of the causal assumptions increases further (Bollen & Pearl, 2013).

As far as the causal assumptions stand, the structural coefficients in the model equations can be used to describe cause-effect sequences (Bentler & Chou, 1987) and one can interpret directional relations in the model as causality (Bollen & Pearl, 2013). Appendix 1 explains the difference in the operational definition between regression coefficients in linear regression models and structural coefficients in structural equation models.

1.6 Misspecifications in Structural Equation Model

As mentioned earlier, model fit (from testable implications) indicates how well the model describes the data. A poor model fit gives us a sign of misspecification happening in

the model. Misspecification happens when i) one or more parameters are free to estimate whose population values are actually zero (over-parameterization), ii) one or more parameters are fixed at zeros and not allowed to estimate freely, yet whose population values are non-zeros (under-parameterization), or iii) both over- and under-parameterizations take place at the same time (Hu & Bentler, 1998). To correct the misspecification, knowing the location of the under- and/or over-parameterization taking place in the model is paramount on a practical level.

Some sources of misspecification in measurement models are i) the number of common factors where the model is assigned either more or less latent factors than it should be (Goretzko & Bühner, 2020), ii) cross-factor loadings (see Asparouhov & Muthén, 2009) and iii) correlated uniquenesses that the item residuals could covary. Two sources of misspecification in path models are i) the number of pathways and ii) pathway directionality. Regarding the number of pathways, a missing pathway is classified as an under-parameterization that the estimate of the structural parameter is fixed at zero or specific value and yet the population value is not; a redundant pathway is classified as an over-parameterization that the structural parameter is freely estimated whose population value is zero. Regarding the pathway directionality, it happens when one assigns that variable *x* causes *y* but the truth is *y* causes *x*. this phenomenon can be comprehended as an under-parameterization and an over-parameterization that happen spontaneously between *x* and *y*. Model fit is intent on alerting one the presence of misspecification. Yet, it gives little or no information as to what kind of misspecification is in the poorly fitted model. The focus of this thesis is on discovering path models but not measurement models, we will look into some ways to identify the path-model misspecifications.

1.7 Ways to Identify Path-Model Misspecifications

1.7.1 Solutions to the Number of Pathways

Since their appearance, global (or traditional) fit has been under criticism that they fall short of adequate sensitivity to misspecification located in the path model (e.g., Anderson & Gerbing, 1988; Mulaik et al., 1989), largely due to the difference in the number of parameters between the path and measurement models. As the number of parameters in the path model is usually disproportionately lower than that of the measurement model, the measurement model dominates the sensitivity of global fit. Hence, regardless the path model is correctly specified or not, global fit primarily reflects the fitness of the measurement model, paying insufficient attention to the fitness of the path model (Mulaik et al., 1989). In respect to the characteristic of global fit that has insufficient sensitivity against the misspecification in path models, some fit indices are aimed at detecting the path-model misspecification which are collectively known as path-related fit indices. Using the sequence of nested models, path-related fit indices emerged in the early 1990s, of which we will introduce two: path-related root mean square error of approximation (RMSEA-P) and path-related confirmatory fit index (CFI-P).

RMSEA-P. RMSEA-P is an absolute fit that presents the degree of misfit in the path model per degree of freedom, or literally how distant a fitted path model is away from its saturated path model (i.e., SS). RMSEA-P features an elevated sensitivity to the misspecification in the path model (McDonald & Ho, 2002), which is specified as:

$$\text{RMSEA-P} = \sqrt{\max\left[\left(\frac{\left(\chi^2_{\text{T}}-\chi^2_{\text{SS}}\right)-\left(df_{\text{T}}-df_{\text{SS}}\right)}{\left(df_{\text{T}}-df_{\text{SS}}\right)\times(N-1)}\right), 0\right]} \tag{1.4}$$

where χ^2_{T} and χ^2_{SS} are chi-square test statistics for the theoretical model (T) and structural saturated model (SS), respectively (see Appendix 2 for their definitions); df_{T} and df_{SS} are their degrees of freedoms ($df_{\text{T}} > df_{\text{SS}}$). If $df_{\text{T}} = df_{\text{SS}}$, RMSEA-P is set to zero.

CFI-P. CFI-P is a relative fit index that presents one minus the ratio of the non-centrality parameters between the fitted path model to its independence path model (i.e.,

structural null model, SN). In other words, it is indicative of the discrepancy in fit between a fitted path model and its SN. As derived from the non-centrality structural covariance index (NSCI; Williams & Holahan, 1994), CFI-P has a heightened sensitivity to the path model misspecification whose expression is:

$$\text{CFI-P} = \min\left[\left(1 - \frac{\left(\chi^2_{\text{T}} - \chi^2_{\text{SS}}\right) - \left(df_{\text{T}} - df_{\text{SS}}\right)}{\left(\chi^2_{\text{SN}} - \chi^2_{\text{SS}}\right) - \left(df_{\text{SN}} - df_{\text{SS}}\right)}\right), 1\right] \tag{1.5}$$

Recommended Cut-off. Despite the methodological soundness of RMSEA-P and CFI-P, few studies were conducted on their recommended criterion values to decide whether a path model is acceptable. Scholars (e.g., Bolkan & Goodboy, 2015; Lance et al., 2016; Williams & O'Boyle, 2011) consistently interpreted the values of RMSEA-P using the guidelines for the traditional RMSEA. According to the interpretation that O'Boyle and Williams (2011) used, a path model with the RMSEA-P value below or equal to 0.05 is described as a close fit, between 0.05 and 0.08 as a reasonable fit, between 0.08 and 0.10 a mediocre fit, and above 0.10 a poor fit. For CFI-P, Lance and colleagues (2016) advise that CFI-P greater than 0.99 is indicative of an acceptable model fit.

1.7.2 Solutions to Pathway Directionality

Incorrectly specifying pathway directionality risks reporting spurious findings, drawing a conflicting conclusion, and misguiding future research (Henley et al., 2006). We will review three ways to identify the directions of pathways in the current literature: elimination approach, direction dependence analysis and SEM with higher-order moment structures.

Elimination Approach. Elimination approach (Henley et al., 2006) is a five-step theoretically based deduction approach that makes use of scientific literatures and substantive theories to alleviate the risk of assigning wrong directions to the effects in a model (Raykov & Marcoulides, 2001; Raykov & Penev, 1999). The five steps are:

1. Being aware of the potential effect of equivalent models,
2. Locating potential equivalent models,
3. Judging whether equivalent models are theoretically plausible,
4. Eliminating some if not all equivalent models via the study design, and;
5. Discussing the limitation of retaining some equivalent models if they cannot be ruled out as alternative interpretation to the study.

Elimination approach is proved to be useful in confirmatory empirical studies with prior studies to refer to. Yet, it is of no use with exploratory studies where researchers have no previous studies to count on.

Direction Dependence Analysis. Direction dependence analysis (DDA; Wiedermann & von Eye, 2015) is a battery of statistical means to validate causal relationships. Unlike elimination approach, DDA demands no prior studies to determine the cause-and-effect sequence. Instead, DDA exploits the asymmetric distributions of measured variables (Dodge & Rousson, 2000; 2001). The underlying assumption in DDA is that, the true cause variable is non-normal distribution and the error term of the true effect variable is in normal distribution. The consequence is that, the true effect variable is less deviated from the normal distribution than the true cause variable does. If the direction is assigned correctly, then the effect variable would have a distribution more normal than the cause variable, so do the error term. If the direction is wrong, then the effect variable would have a distribution more deviated from normal than the cause variable does; the error term would inherit the non-normality from the respective effect variable as well. Apart from the first-order moment (i.e., mean vector) and the second-order moment (covariance matrix), DDA takes into account the third-order moment (skewness) and the fourth-order moment (kurtosis) when it comes to measuring and comparing the non-normality between two variables and between the

corresponding error terms from least square regressions. The skewness for a variable x (denoted by ζ_x) and the kurtosis (denoted by δ_x) are expressed as:

$$\zeta_x = E\left[\left(x-\mu_x\right)^3\right] \tag{1.6}$$

$$\delta_x = E\left[\left(x-\mu_x\right)^4\right] \tag{1.7}$$

where "E" denotes the expectation operator and μ_x denotes the mean value of x. When x is regressed on y, we obtain from the error term of x, $e_x^{(y)}$. The skewness for $e_x^{(y)}$ (denoted by ζ_{exy}) and the kurtosis (denoted by δ_{exy}) are:

$$\zeta_{exy} = \left(1-\rho_{xy}^2\right)^{\frac{3}{2}} \zeta_x \tag{1.8}$$

$$\delta_{exy} = \left(1-\rho_{xy}^2\right)^{2} \delta_x \tag{1.9}$$

where ρ_{xy}^2 denotes the Pearson correlation between x and y. According to the decision rules for the measured variables set by DDA, if ζ_x is greater (smaller) than ζ_y and δ_x is greater (smaller) than δ_y in absolute values, it suggests that x (y) is the cause (Wiedermann & Sebastian, 2020). Regarding the decision rules for the error terms, if ζ_{exy} is greater (smaller) than ζ_{eyx} and δ_{exy} is greater (smaller) than δ_{eyx}, it suggests that x (y) is the cause (Wiedermann & Sebastian, 2020). It is possible that the decisions drawn from the variables and from the error terms are inconsistent.

In addition to the two scenarios that i) x causes y and ii) y causes x, DDA considers the third that a latent confounding variable causes both x and y. To detect the presence of latent confounding variables, it studies the independence between the cause variable and the error term which is quantified by Hilbert-Schmidt independence criterion (Gretton et al., 2007) and Brownian distance correlation (Székely et al., 2007). If x and $e_x^{(y)}$ are not independent. and y and $e_y^{(x)}$ are not independent, it infers the presence of latent confounding variables (Wiedermann & Sebastian, 2020).

Lately, DDA has been adapted to latent-variable models (e.g., Pollaris & Bontempi, 2020; von Eye & Wiedermann, 2014) where the true scores on latent variables are approximated by their factor scores computed from the results of principal component analysis (PCA). The performance of DDA in both observed-variable and latent-variable contexts appears promising, with its implementation on SPSS (Wiedermann & Li, 2018) and R being available for researchers. Sadly, DDA is designed to test two variables at a time. For more than two variables, transitivity is necessary. Besides, statistical tests in DDA may give contradictory results (e.g., Wiedermann & Sebastian, 2020) and are not warranted to yield consistent conclusions.

Structural Equation Modelling with Higher-Order Moment Structures. Standard SEM harnesses the first-order and the second-order moments in model estimation while it does not gather information from the third- and fourth-order moments. That is mainly because the two higher-order moments add nearly no information on top of the first- and second-order moments. Another reason is that the higher-order moments can be computationally demanding. Therefore, SEM safely assumes multivariate normality.

However, the multivariate normality assumption imposes limitations on SEM. For example, estimation of non-linear effects requires information beyond the first- and second-order moments (Mooijaart & Bentler, 2010; Mooijaart & Satorra, 2012). To determine the directions of effects, Shimizu and Kano (2008) propose a variant of SEM that capitalizes on higher-order moment structures which is dubbed as “non-normal SEM”. In non-normal SEM, asymptotically distribution-free method (Browne, 1984) is responsible for the parameter estimation. Fit statistics obtained from non-normal SEM enable researchers to find out the effect directions. Despite its appealing feature, non-normal SEM managed to attract little usage in empirical studies (e.g., Takahashi et al., 2012). At least three reasons hinder non-normal SEM from broad applications. First, the risk of encountering a non-identifiable model

increases as additional unknown parameters are needed in the estimation of higher-order moment structures (Mooijaart, 1985). Second, non-normal SEM demands a sufficiently large sample size (Mooijaart & Satorra, 2012). Taking asymptotically distribution-free approach as an estimation procedure, non-normal SEM demands a sample size of 5,000 to generate reliable parameter estimates for a factor-analysis model with 15 observed variables and three latent variables (Yuan & Bentler, 1998). Asymptotically distribution-free approach is thus not recommended when the sample size is below the number of unique, non-duplicated elements of the sample covariance (Bentler & Yuan, 1999). Last, neither commercial nor open-source statistical software is compatible with non-normal SEM. To the best of our knowledge, EQS (version 7 or newer; Mooijaart & Bentler, 2010) is the only software that implements SEM with selected third-order moments for latent interactions.

Chapter 2: Literature Review of Existing Causal Discovery Methods

Causal discovery methods are a collection of algorithms that aims to infer the causal relationship. The input is mainly a raw data. The output is a discovered model which is usually presented in the form of DAG. The goal is to derive a causal graph portraying the causal associations as close to the true DAG as possible. The reason of preferring DAG over other graphs is the capability of presenting conditional independence assumptions (Pearl, 1995). In the following, we will briefly review two dominant forces of causal discovery methods: Gaussian causal discovery methods and non-Gaussian causal discovery methods.

2.1 Gaussian Causal Discovery

PC algorithm, fast causal inference algorithm and greedy equivalence search algorithm are some Gaussian causal discovery methods that assume CMC, CFC and linearity (where variables are linear functions of constants and other variables), not least Gaussian distribution.

2.1.1 PC Algorithm

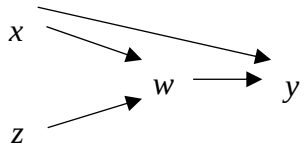
PC algorithm, named after Peter Spirtes and Clark Glymour (1991), is a constraint-based algorithm to offer a skeleton or a search architecture. With the aid of PC algorithm, other Gaussian discovery algorithms are able to decide the conditional independence of variables and discover a causal diagram. PC algorithm does not assume the presence of latent confounding variables. Assume four variables (x, z, w and y) whose true causal graph is shown in Figure 5a. The operation of PC algorithm is as below.

1. Construct a fully completed undirected graph where undirected edges are added to link every pair of variables possible. A graph with four variables constitutes a maximum of six undirected edges (see Figure 5b).
2. Prune undirected edges between variables if they are (conditionally) independent when taking other variable(s) into account. In Figure 5c, for example, an undirected

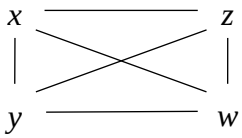
edge between *x* and *z* is eliminated because $x \perp\!\!\!\perp z$; an undirected edge between *y* and *z* is eliminated as $y \perp\!\!\!\perp z \mid x$.

3. Assign directions to undirected edges according to some orientations, such as v-structure. For example, with the orientation of v-structure, add arrowheads to the path *x*–*w*–*z* to become $x \rightarrow w \leftarrow z$.

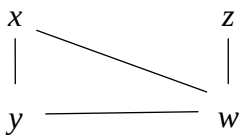
Figure 5 *Illustration of Causal Discovery with PC Algorithm*



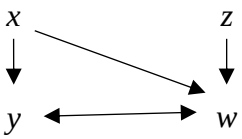
5a) True causal diagram.



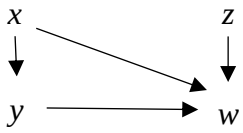
5b) A fully-connected undirected graph to begin with.

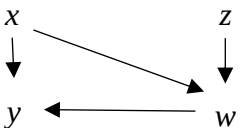


5c) Some undirected edges are pruned because of independence between some variables.



5d) Directionality added to some undirected edges.





5e) Two possible DAGs

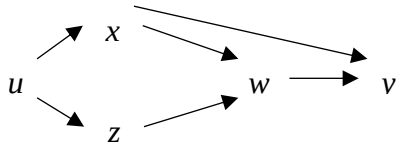
After running PC algorithm, directions of some edges remain unknown where arrowheads are possible at either end, such as edge *y*–*w*. Keep in mind that these edges are not undirected edges but edges with undecided directions. By convention, they are temporarily represented by bidirectional edges (see Figure 5d). Due to the presence of edges with unknown directions, PC algorithm produces a set of possible DAGs, rather than one definite DAG. Back to the

example, an edge with undecided direction gives two possible DAGs (see Figure 5e). See Glymour et al. (2019) for the details of PC algorithm.

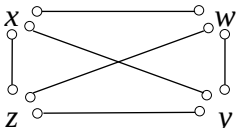
2.1.2 Fast Causal Inference Algorithm

Fast causal inference (FCI) algorithm (Spirtes et al., 2000) is a constraint-based algorithm whose merit is the generalizability to causal graphs with latent confounding variables. FCI algorithm can accommodate the presence of latent confounders and locate them to some extent. The output of FCI algorithm does not necessarily conform to DAG as it allows the presence of latent confounding variables in violation of DAG assumptions.

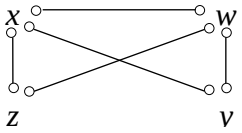
Figure 6 *Illustration of Causal Discovery with FCI Algorithm*



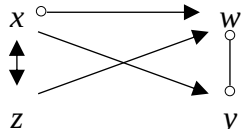
6a) True causal diagram



6b) A fully-connected partial ancestral graph to begin with



6c) Some bidirected edges are pruned because of independence between some variables



6d) Directionality added to some undirected edges.

Let Figure 6a be the true causal diagram of four variables (*x*, *w*, *y* and z) with *u* being a latent confounder. Below illustrates how FCI algorithm operates:

1. Construct a complete undirected graph where the “o” mark at the end of edges indicate that it can either be an arrowhead or an arrow tail (see Figure 6b).
2. Prune edges between variables if they are independent, either unconditionally or conditionally (see Figure 6c).
3. Assign directions to undirected edges based on some orientations (e.g., v-structure).

In Figure 6d, the bidirectional edge between *x* and *z* flags the presence of latent confounders. The remaining "o" marks suggest that FCI algorithm cannot decide whether an arrowhead or an arrow tail is best in the position. For example, the edge between *w* and *y* can be either a directed edge extending from *w* to *y*, a directed edge from *y* to *w*, or a latent confounding variable linking to both *w* and *y*.

2.1.3 Greedy Equivalence Search Algorithm

Greedy equivalence search (GES) algorithm (Chickering, 2002) is a score-based algorithm that aims to maximize some quasi-Bayesian score function, typically BIC, whilst the previous two algorithms are based on conditional independence tests. GES algorithm is summarized in brief by two phases:

1. In the forward phase, GES algorithm substantiates an empty graph and incrementally adds directed edges to it which improves the score most. The graph complexity rises in the meantime. The forward stage is halted if no improvement in the score is possible by adding more directed edges.
2. In the backward phase, GES algorithm gradually eliminates existing directed edges that most improves the score. The graph becomes less and less complex in the meantime. The backward phase will halt once no further improvement in the score can be made.

Like PC algorithm, GES algorithm entails an assumption of no latent confounders in the true causal graphs.

2.1.4 Issues with Gaussian Causal Discovery Methods

In these Gaussian causal discovery algorithms, observed variables are presumed in Gaussian distribution. They take advantage of the *d*-separation properties to determine the conditional independence between observed variables. However, it is possible that multiple causal graphs share the same *d*-separation properties. As a result, rather than a definite DAG,

these algorithms predict a set of possible DAGs, or a group of "Markov equivalent" causal graphs, all of which possess the identical set of conditional independences.

More information is needed to narrow down the number of the predicted causal graphs. As reviewed earlier, understanding the properties of the asymmetry between variables enables researchers to resolve the cause-effect issue (Glymour et al., 2019; Wiedermann & von Eye, 2015) and assign directions to those edges with undecided directions. Two statistical tools discussed previously (DDA and non-normal SEM) unanimously contend that the Gaussianity assumption obstructs algorithms from accessing information relevant to causality from the third- and fourth-order moments. DDA and SEM with higher-order moments shed the light into the possibility that abandoning the assumption of Gaussian distribution empowers the causal discovery algorithms in the way that they can deduce one causal diagram from the given data, rather than multiple "Markov-equivalent" causal diagrams.

2.2 Non-Gaussian Causal Discovery

Non-Gaussianity is viewed as a preferred way to describe empirical, non-synthetic data than normal distribution (Bono et al., 2017; Micceri, 1989). In one analysis, 5% of data from empirical research relating to psychology and education had approximate normal distributions (Blanca et al., 2013). Non-Gaussian causal discovery makes use of the information including those from higher-order moment structures to determine the cause-effect sequences of variables. The representative is linear non-Gaussian acyclic model (LiNGAM). LiNGAM (Shimizu et al., 2006) is a non-Gaussian variant of SEM and Bayesian networking for causal inference. Four properties of LiNGAM are linearity, non-Gaussianity, directedness (all connections between variables are unidirectional) and acyclicity. In acyclicity, one can sequence all variables in the system in a causal order where ancestors (i.e., earlier variables) cause descendants (later variables); yet descendants never lead to ancestors

and loops are forbidden. Unlike the Gaussian causal discovery algorithms, LiNGAM algorithms output one causal graph in the form of DAG (Pearl, 2000; Spirtes et al., 2000) in which the conditional independences of observed variables are governed by CMC but not CFC (Zhang & Spirtes, 2016). That is, it does not assume that any two observed variables which are not *d*-separated must be dependent. The independence of two observed variables is tested in spite of no *d*-separation between. Also, the discovered causal graph is said to be causally sufficient (Spirtes et al., 2000), meaning that all common causes of observed variables are taken into account in the graph. Hidden variables outside of the model are not presumably a source of observed variables in the causal graph.

In the data generation process, suppose **x** is a ($p \times 1$) random vector of observed variables, with a set of the observed variable subscripts as $U = \{1, \ldots, p\}$. Without loss of generality, observed variables in **x** have zero means. In LiNGAM, the linear function of an observed variable x_i , for all $i \in U$, is specified as:

$$x_i = \sum_{k(j)<k(i)} \gamma_{ij} x_j + e_i \tag{2.1}$$

where $k(i)$ denotes the causal order of x_i. In equation 2.1, the expression $k(j) < k(i)$ means that x_i is regressed on every x_j as long as the causal order of x_j is ahead of x_i. The linear relation of **x** in a matrix form is expressed as:

$$\mathbf{x} = \mathbf{\Gamma x} + \mathbf{e} \tag{2.2}$$

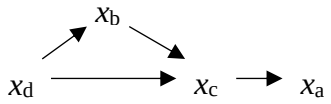
where $\mathbf{\Gamma}$ is a ($p \times p$) regression coefficient matrix and **e** is a ($p \times 1$) random vector of error terms. Due to acyclicity, $\mathbf{\Gamma}$ is permutable to strict lower triangularity. Strict lower triangularity refers to a feature of the matrix which has zero-value elements above and on the diagonal. **e** is assumed to be in non-Gaussian distribution and are independent:

$$\mathrm{pr}(e_1,\ldots, e_\mathrm{p}) = \prod_{i=1}^{p} \mathrm{pr}(e_i) \tag{2.3}$$

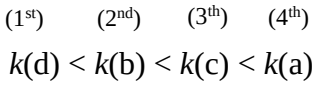
where "pr" denotes the density function of the joint probability distribution. From equation 2.3, the orthogonality of error terms alludes to the absence of latent confounding variables in a DAG (Spirtes et al., 1993).

Figure 7 illustrates the operational flow of LiNGAM with four variables (x_a, x_b, x_c, x_d). The true causal diagram is in Figure 7a and the causal order (denoted by K) would be $k(d) < k(b) < k(c) < k(a)$. According to K, x_d is the first variable (or the variable with no parent); x_a is the last variable (or the variable with no child). x_d is an ancestor of x_b since $k(d) < k(b)$. To derive the $\mathbf{\Gamma}$ with the dimension of (4 × 4) from the data, LiNGAM algorithms first sequence the four variables in a causal order (see Figure 7b). With list K, the algorithms use regression with regularization to derive a $\mathbf{\Gamma}$ in strict lower triangularity (Bollen, 1989; see Figure 7c) which can be illustrated as a DAG (see Figure 7d). We will continue with the example of the four observed variables (x_a, x_b, x_c, x_d), or the four-variable case, for the purpose of illustration in the remaining of Chapter 2.

Figure 7 *Illustration of Causal Discovery with LiNGAM Algorithm*



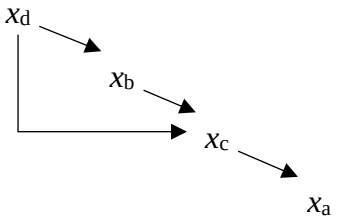
7a) True causal diagram



7b) Determine the causal order

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ \gamma_{bd} & 0 & 0 & 0 \\ \gamma_{cd} & \gamma_{cb} & 0 & 0 \\ 0 & 0 & \gamma_{ac} & 0 \end{pmatrix}$$

7c) Causal order constructs a strictly lower triangular regression coefficient matrix



7d) Some directed edges are pruned using regression with regularization

Over the past few decades, the development of LiNGAM algorithms has been in proliferation. Some are designed to tackle conventional regression with observed variables,

including ICA-LiNGAM (Shimizu et al., 2006), Pairwise LvLiNGAM algorithm (Entner & Hoyer, 2011), DirectLiNGAM (Shimizu et al., 2011) and ParceLiNGAM (Tashiro et al., 2014). More algorithms advance to tackle complex models such as longitudinal models (Hyvärinen et al., 2010; Moneta et al., 2013), those with latent confounding variables (e.g., Maeda & Shimizu, 2020; Mooij et al., 2009) and non-linear models (Hoyer et al., 2008). This thesis examines three basic algorithms: ICA-LiNGAM, DirectLiNGAM and ParceLiNGAM. Each has their unique mechanism to sequence variables into a causal order. Pairwise LvLiNGAM algorithm is not considered here as it cannot handle more than two variables without transitivity and does not estimate regression coefficients between variables.

2.2.1 ICA-LiNGAM for Observed Variables

Independent component analysis-based LiNGAM (ICA-LiNGAM; Shimizu et al., 2006) is a causal inference algorithm that identifies LiNGAM for observed variables with independent component analysis (ICA). ICA (Jutten & Herault, 1991) is an unsupervised method that involves the linear decomposition of multivariate non-gaussian data into components that are statistically independent (Hyvärinen & Smith, 2013). ICA-LiNGAM involves three steps: i) identification of LiNGAM with ICA, ii) permutation of unmixing matrix to strict lower triangularity, and iii) adaptive lasso regression.

Step 1: Identification of LiNGAM with ICA. The reduced form of equation 2.2 about the linear functions of observed variables is:

$$\mathbf{x} = (\mathbf{I} - \mathbf{\Gamma})^{-1}\mathbf{e} \tag{2.4}$$

where **I** is an identity matrix and the superscript "–1" refers to the inverse of a matrix. Equation 2.4 fits the definitions of a standard ICA model. ICA is aimed at separating observed variables into independent error terms with reference to equation 2.4. Suppose **s** is a random vector of ($h \times 1$) independent components where $a \leq p$. A standard linear ICA model is expressed as:

$$\mathbf{x} = \mathbf{A}_{\text{ica}}\mathbf{s} \tag{2.5}$$

where $\mathbf{A}_{\text{ica}}$ is a ($p \times h$) mixing matrix estimated by ICA. To identify a LiNGAM with ICA, we restrict the number of independent components to be equal to that of the observed variables (i.e., $h = p$). Subsequently, $\mathbf{A}_{\text{ica}}$ becomes a squared matrix with a full column rank. The inverse of $\mathbf{A}_{\text{ica}}$ is a ($h \times p$) unmixing matrix $\mathbf{W}_{\text{ica}}$ (i.e., $\mathbf{A}_{\text{ica}} = \mathbf{W}_{\text{ica}}^{-1}$), with which one can rewrite equation 2.5:

$$\mathbf{x} = \mathbf{W}_{\text{ica}}^{-1}\mathbf{s} \tag{2.6}$$

At this stage, we are unable to resolve equation 2.4 with equation 2.6 with the relation that $\mathbf{W}_{\text{ica}}$ equates to ($\mathbf{I} - \mathbf{\Gamma}$), due to two indeterminacies one obtains from ICA: permutation indeterminacy and scaling indeterminacy. Permutation indeterminacy is an uncertainty that the sequence of independent components in **s** is arbitrary and unknown; thus, the order of independent components is not warranted to be in line with the order of observed variables in **x** (Comon, 1994; Hyvärinen & Oja, 2000). Scaling indeterminacy is an uncertainty that the values of variances in independent components are unknown (Hyvärinen & Oja, 2000). We will look into the two indeterminacies in detail as follows.

To have a clear view of permutation indeterminacy, we consider a vector of observed variable as $\mathbf{x} = (x_{\text{a}}, x_{\text{b}}, x_{\text{c}}, x_{\text{d}})^{\text{T}}$. Under the effect of permutation indeterminacy, the sequence of **s** resulted from ICA is unknown, meaning that **s** can be either $(x_{\text{d}}, x_{\text{c}}, x_{\text{b}}, x_{\text{a}})^{\text{T}}$ or $(x_{\text{b}}, x_{\text{d}}, x_{\text{a}}, x_{\text{c}})^{\text{T}}$ or alike. What's more, the variable sequences in the row of $\mathbf{A}_{\text{ica}}$ and in the column of $\mathbf{W}_{\text{ica}}$ are unknown as well. To illustrate the undesirable impact of having an unknown sequence in **s**, suppose an ideal mixing matrix as **A** for four observed variables:

$$\begin{pmatrix} a_{\text{aa}} & a_{\text{ab}} & a_{\text{ac}} & a_{\text{ad}} \\ a_{\text{ba}} & a_{\text{bb}} & a_{\text{bc}} & a_{\text{bd}} \\ a_{\text{ca}} & a_{\text{cb}} & a_{\text{cc}} & a_{\text{cd}} \\ a_{\text{da}} & a_{\text{db}} & a_{\text{dc}} & a_{\text{dd}} \end{pmatrix}$$

where element a_{ij} describes the relation between x_i and x_j, for $i, j = \{a, b, c, d\}$. In the target **A**, the row corresponds to the column. Because of permutation indeterminacy, one of the many possible $\mathbf{A}_{ica}$ looks like this.

$$\begin{pmatrix} a_{ad} & a_{ac} & a_{ab} & a_{aa} \\ a_{bd} & a_{bc} & a_{bb} & a_{ba} \\ a_{cd} & a_{cc} & a_{cb} & a_{ca} \\ a_{dd} & a_{dc} & a_{db} & a_{da} \end{pmatrix}$$

As a result, we have no idea as to whether the row of $\mathbf{A}_{ica}$ corresponds to its column. Thus, $\mathbf{A}_{ica}$ is not identified. The same applies to $\mathbf{W}_{ica}$.

A mathematical solution to both permutation indeterminacy and scaling indeterminacy is:

$$\mathbf{W}_{ica} = \mathbf{P}\mathbf{D}\widetilde{\mathbf{W}}'_{ica} \tag{2.7}$$

where **P** is a ($h \times p$) permutation matrix, **D** is a ($p \times p$) diagonal matrix, and $\widetilde{\mathbf{W}}'_{ica}$ is a ($p \times p$) unmixing matrix being corrected with its column order corresponding to its row order. In other word, $\widetilde{\mathbf{W}}'_{ica}$ is free of permutation and scaling indeterminacies. See Appendix 3 for the way to derive the solution shown in equation 2.7. Substituting $\widetilde{\mathbf{W}}'_{ica}$ into the relation that $\mathbf{W} = \mathbf{I} - \mathbf{\Gamma}$, one has:

$$\widetilde{\mathbf{W}}'_{ica} = \mathbf{I} - \mathbf{\Gamma}_{ica} \tag{2.8}$$

where $\mathbf{\Gamma}_{ica}$ is a ($p \times p$) adjacency matrix that has row and column orders corresponding to the order of observed variables in **x**. However, one should not confuse $\mathbf{\Gamma}_{ica}$ with $\mathbf{\Gamma}$ as $\mathbf{\Gamma}_{ica}$ is not yet to be permuted to strict lower triangularity.

Step 2: Permutation to Strict Lower Triangularity. The second step of ICA-LiNGAM algorithm is to permute the unmixing matrix $\mathbf{\Gamma}_{ica}$ obtained from equation 2.8 to strict lower triangularity so as to match the definition of $\mathbf{\Gamma}$ in LiNGAM. The permutation of the unmixing matrix $\mathbf{\Gamma}_{ica}$ is expressed in equation as:

$$\widetilde{\mathbf{\Gamma}}_{ica} = \widetilde{\mathbf{P}}\, \mathbf{\Gamma}_{ica} \widetilde{\mathbf{P}}^{T} \tag{2.9}$$

where $\widetilde{\mathbf{P}}$ is a ($p \times p$) unknown permutation matrix which differs from **P** in equation 2.7, and $\widetilde{\mathbf{P}}^{\mathrm{T}}$ is the transpose of $\widetilde{\mathbf{P}}$. As a result, $\tilde{\mathbf{\Gamma}}_{\mathrm{ica}}$ is a ($p \times p$) unmixing matrix in strict lower triangularity. In practice when $|p| > 2$, searching a $\widetilde{\mathbf{P}}$ can be computationally demanding and sometimes may not be feasible. As such, Shimizu et al. (2011) propose a method to approximate the permutation to ease the computation which involves repeated removal of elements in $\mathbf{\Gamma}_{\mathrm{ica}}$ and repeated testing. Appendix 4 proves details of the said method. Once the permutation on $\mathbf{\Gamma}_{\mathrm{ica}}$ to strict lower triangularity is completed, we obtain a complete list *K* that describes the causal order of observed variables in **x**.

Step 3: Adaptive Lasso Regression. In the causal order as derived in Step 2, earlier variables may or may not have direct effect on later variables, since being an ancestor is a necessary yet not sufficient condition of being a parent. To see if the direct effect of ancestors on their descendents is present or not, we take adaptive least absolute shrinkage and selection operator lasso regression (Tibshirani, 1996; Zou, 2006) or adaptive lasso regression in short. Adaptive lasso regression features a regularization technique that selects a model with high interpretability and at the same time maintains an optimal prediction of parameter estimates. The estimator of adaptive lasso regression is specified as (Zou, 2006):

$$\hat{\boldsymbol{\gamma}} = \arg\min_{\boldsymbol{\beta}} \left| x_i - \sum_{j=1}^{p-1} \gamma_{ij} x_j \right|^2 + \lambda \sum_{j=1}^{p-1} w_{ij} |\gamma_{ij}| \qquad (2.10)$$

where λ denotes the regularization parameter and w_{ij} denotes the data-driven weight term corresponding to x_i and x_j.

LiNGAM algorithms make use of adaptive lasso regressions to derive $\mathbf{\Gamma}$ from the causal order of observed variables. Specifically, a series of adaptive lasso regressions are performed in which regressors and outcome variables are assigned in accordance with the causal order. To illustrate, we present how to run adaptive lasso regression on the four-variable case used earlier. Suppose we have a complete causal order as *K* (see Figure 7b) and

substantiate $\mathbf{\Gamma} = \{\gamma_{ij}\}$ as a (4 × 4) matrix for $i, j = \{a, b, c, d\}$. Since an observed variable is prohibited to have a direct effect onto itself, we set γ_{ii} to zero and the $\mathbf{\Gamma}$ becomes:

$$\mathbf{\Gamma} = \begin{pmatrix} \gamma_{aa} & \gamma_{ab} & \gamma_{ac} & \gamma_{ad} \\ \gamma_{ba} & \gamma_{bb} & \gamma_{bc} & \gamma_{bd} \\ \gamma_{ca} & \gamma_{cb} & \gamma_{cc} & \gamma_{cd} \\ \gamma_{da} & \gamma_{db} & \gamma_{dc} & \gamma_{dd} \end{pmatrix} = \begin{pmatrix} 0 & \gamma_{ab} & \gamma_{ac} & \gamma_{ad} \\ \gamma_{ba} & 0 & \gamma_{bc} & \gamma_{bd} \\ \gamma_{ca} & \gamma_{cb} & 0 & \gamma_{cd} \\ \gamma_{da} & \gamma_{db} & \gamma_{dc} & 0 \end{pmatrix}$$

In the given K, the first variable x_d is said to be the true source and thus no regression is run on it. We can safely set all elements in the fourth row of $\mathbf{\Gamma}$ to zeros.

$$\mathbf{\Gamma} = \begin{pmatrix} 0 & \gamma_{ab} & \gamma_{ac} & \gamma_{ad} \\ \gamma_{ba} & 0 & \gamma_{bc} & \gamma_{bd} \\ \gamma_{ca} & \gamma_{cb} & 0 & \gamma_{cd} \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Provided in K that $k(d) < k(b)$, to find out if x_d has a direct effect on x_b, we run an adaptive lasso regression and replace γ_{bd} in $\mathbf{\Gamma}$ with the obtained regression parameter ($\bar{\gamma}_{bd}$). As x_a and x_c are children of x_b, we set γ_{ba} and γ_{bc} to zeros in the second row of $\mathbf{\Gamma}$. $\mathbf{\Gamma}$ now becomes:

$$\mathbf{\Gamma} = \begin{pmatrix} 0 & \gamma_{ab} & \gamma_{ac} & \gamma_{ad} \\ 0 & 0 & 0 & \bar{\gamma}_{bd} \\ \gamma_{ca} & \gamma_{cb} & 0 & \gamma_{cd} \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

We move to the third variable in K, x_c. Since $k(d) < k(b) < k(c)$, we are interested in whether x_d and x_b each has a direct effect on x_c. We conduct adaptive lasso regressions where x_c is regressed on x_d and x_b. The two obtained regression parameters, $\bar{\gamma}_{cd}$ and $\bar{\gamma}_{cb}$, replace γ_{cb} and γ_{cd} in $\mathbf{\Gamma}$. At the same time, γ_{ca} is set to zero as x_a is a child of x_c and will never have a direct effect to its parent. Now, we renew $\mathbf{\Gamma}$ to:

$$\mathbf{\Gamma} = \begin{pmatrix} 0 & \gamma_{ab} & \gamma_{ac} & \gamma_{ad} \\ 0 & 0 & 0 & \bar{\gamma}_{bd} \\ 0 & \bar{\gamma}_{cb} & 0 & \bar{\gamma}_{cd} \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

At last, the fourth and last variable, x_a, is the child of all the other three observed variables. Again with adaptive lasso regression, we regress x_a on x_d, x_b and x_c, from which the obtained regression parameters ($\bar{\gamma}_{ad}$, $\bar{\gamma}_{ab}$ & $\bar{\gamma}_{ac}$) replace those in $\mathbf{\Gamma}$.

$$\mathbf{\Gamma} = \begin{pmatrix} 0 & \bar{\gamma}_{ab} & \bar{\gamma}_{ac} & \bar{\gamma}_{ad} \\ 0 & 0 & 0 & \bar{\gamma}_{bd} \\ 0 & \bar{\gamma}_{cb} & 0 & \bar{\gamma}_{cd} \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Because of regularization in adaptive lasso regression, it is possible that the estimate of either $\bar{\gamma}_{ab}$, $\bar{\gamma}_{ac}$, $\bar{\gamma}_{ad}$, $\bar{\gamma}_{bd}$, $\bar{\gamma}_{cb}$ and/or $\bar{\gamma}_{cd}$ is penalized to zero. After running adaptive lasso regression, the finalized $\mathbf{\Gamma}$ can be permuted in strict lower triangular **(**Bollen, 1989):

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ \bar{\gamma}_{bd} & 0 & 0 & 0 \\ \bar{\gamma}_{cd} & \bar{\gamma}_{cb} & 0 & 0 \\ \bar{\gamma}_{ad} & \bar{\gamma}_{ab} & \bar{\gamma}_{ac} & 0 \end{pmatrix}$$

Table 1 gives the algorithm overview of ICA-LiNGAM. For those who are interested in the technical details of the algorithm, please visit Appendix 6. ICA-LiNGAM inherits issues of ICA. Even though ICA always achieves global maxima and yields an optimum solution in separating two unknown sources, it is unreliable when separating more than two (Comon, 1994). It infers that, for models with more than two observed variables, ICA-LiNGAM does not guarantee to generate the best possible solution but may deduce incorrect causal orders of variables. In response to the concern, some other LiNGAM algorithms choose not to use ICA in model identification.

Table 1 *Algorithm Overview of ICA-LiNGAM*

Input: Data matrix
Step 1: Estimate an unmixing matrix with independent component analysis, where the number of independent components is set to the number of observed variables.
Step 2: Permute the unmixing matrix to the one filled with non-zero diagonal elements.
Step 3: Standardize the permuted unmixing matrix.
Step 4: Obtain the causal order of observed variables.
Step 5: Estimate a matrix of regression coefficients with adaptive lasso regression.
Output: A matrix of regression coefficients between observed variables

2.2.2 DirectLiNGAM for Observed Variables

DirectLiNGAM (Shimizu et al., 2011) is a causal discovery algorithm that identifies LiNGAM for observed variables using least square regressions on pairs of observed variables. DirectLiNGAM is a top-down approach in discovering the causal order of observed variables in the sense that it begins by finding the first variable and ends by finding the last variable. When one variable is determined, its effect on the rest of the variables is removed. Once the causal order is complete, it follows with adaptive lasso regressions. DirectLiNGAM endorses the difference in mutual information as a statistical independence measure (Hyvärinen, 1998).

Determination of the First Variable. Again, suppose **x** as a ($p \times 1$) random vector of observed variables with the corresponding set of observed variable subscripts as U, $\{1, \ldots, p\}$. Define **x**' as a ($p' \times 1$) random vector of observed variables which is a duplicate of **x** (i.e., $\mathbf{x}' \coloneqq \mathbf{x}$) and define $U' = \{1, \ldots, p'\}$ as the respective set of observed variable subscripts. To complete a causal order of observed variables as K, DirectLiNGAM begins with identifying which is the cause and which is the effect between a pair of observed variables in **x**'.

In a scenario where we desire to find out the cause between two observed variables (x_i' and x_j') in **x**', we regress x_i' on x_j' which gives error term $e_i^{(j)}$; we regress x_j' on x_i' which gives error term $e_j^{(i)}$. According to Shimizu et al. (2011), x_j' is exogenous if and only if x_i' is independent of $e_j^{(i)}$. The implication is that, when the independence between x_i' and $e_j^{(i)}$ is greater than the independence between x_j' and $e_i^{(j)}$, it suggests that x_i' is the true cause, not x_j'. As independence is not the same as correlation (Darroch & James, 1974), we need an independence measure to monitor the level of independence which is mutual information (Hyvärinen, 1998). Mutual information is viewed as the amount of information mutually shared between two variables in the units of differential entropy. Let the mutual information between variables *x* and *y* be MI(*x*, *y*). Surrounding the two regressions about x_i' and x_j', the difference in mutual information (denoted by ΔMI) can be expressed as:

$$\Delta\text{MI} = \text{MI}\left(x_j', e_i^{(j)}\right) - \text{MI}\left(x_i', e_j^{(i)}\right) = H\left(x_j'\right) + H\left(e_i^{(j)}\right) - \left(H\left(x_i'\right) + H\left(e_j^{(i)}\right)\right) \tag{2.11}$$

where var(*u*) and *H*(*u*) are the variance in and the one-dimensional differential entropy of variable *u*. As shown in equation 2.11, ΔMI is a function of one-dimensional differential entropies, for the proof of which please see Appendix 5. Since one-dimensional differential entropy does not have an explicit density function, Hyvärinen (1998) introduces an approximation:

$$H(u) = \frac{1}{2}(1+\log 2\pi) - k_1[E\{\log\cosh(u)\} - \gamma]^2 - k_2\left[E\left\{(u)\exp\left(-\frac{u^2}{2}\right)\right\}\right]^2 \tag{2.12}$$

where constants $k_1 \approx 79.047$ and $k_2 \approx 7.413$ are derived from the Taylor approximation of logarithmic function; $\gamma \approx 0.375$ is the expected value of log cosh in a standardized Gaussian distribution. In equation 2.12, the first term represents the entropy of the standardized Gaussian distribution; the second and third terms combined in absolute term are measures of non-Gaussianity. *H*(*u*) achieves its maximum value when variable *u* is in a perfect Gaussian

distribution. The sign of ΔMI indicates which variable between $x_i^{'}$ and $x_j^{'}$ is exogenous. If ΔMI > 0 (ΔMI < 0), the mutual information carried between $x_j^{'}$ and $e_i^{(j)}$ is greater (smaller) than that of $x_i^{'}$ and $e_j^{(i)}$, meaning the level of independence between $x_j^{'}$ and $e_i^{(j)}$ is lower (higher) than that of $x_i^{'}$ and $e_j^{(i)}$. It leads to a conclusion that $x_i^{'}$ ($x_j^{'}$)is the cause.

Up till now, we have considered a pair of observed variables. Given more than two observed variables in **x**' (i.e., $p' > 2$), we have more than one ΔMIs. Hyvärinen and Smith (2013) propose a *M* criterion that gathers multiple ΔMIs and search the exogenous observed variable in **x**'. The *M* criterion for an observed variable, say, $x_i^{'}$, can be expressed as:

$$M(x_i'; U' \backslash i) = \sum_{j' \neq i', j'=1}^{p'} \min\{0, \Delta\mathrm{MI}\}^2$$

$$= \sum_{j' \neq i', j'=1}^{p'} \min\left\{0, \mathrm{MI}\left(x_j', e_i^{(j)}\right) - \mathrm{MI}\left(x_i', e_j^{(i)}\right)\right\}^2 \qquad (2.13)$$

where $U' \setminus i$ refers to the set of observed variable subscripts (U') excluding subscript *i* and "min" is the minimization function. The possible range of the *M* criterion is from zero to positive infinite. To ease the interpretation of the *M* criterion, we removed the constant of negative one at the front from the original equation in Hyvärinen and Smith (2013). If the *M* criterion approaches positive infinite, it means that, when $x_i^{'}$ is paired against each of the $x_j^{'}$, the respective ΔMI is greater than zero at all times and $x_i^{'}$ is always the cause of $x_j^{'}$. As a result, $x_i^{'}$ is regarded as the exogenous variable in **x**'. On the contrary, the more the *M* criterion approaches zero, the more likely the ΔMI is smaller than zero and the less likely $x_i^{'}$ is the exogenous variable in **x**'. Of observed variable in **x**', one with the maximum *M* criterion, say $x_m^{'}$, qualifies for the first variable.

Removing the Effect of the First Observed Variable from the Rest. Let $x_m^{'}$ in **x**' be the first variable from the *M* criterion. We let *K* be an empty list (i.e., $K \coloneqq \emptyset$). The variable

subscript of $x_m^{'}$, m, enters the tail of K. Before searching for the next (second) variable in the causal order, it is essential to isolate the unwanted effect of $x_m^{'}$ from the rest of observed variables in **x'**. One way is to run least squares regressions of $x_m^{'}$ on $x_i^{'}$ (i = 1, 2, …, p' for all $i \neq m$) and retrieves the error term as $e_i^{(m)}$:

$$e_i^{(m)} = x_i^{'} - \bar{\gamma}_{im} x_m^{'}$$

where $\bar{\gamma}_{im}$ is the estimate of the regression coefficient of $x_i^{'}$ on $x_m^{'}$. According to Lemma 2 in Shimizu et al. (2011), if one substitutes $e_i^{(m)}$ for $x_i^{'}$, a LiNGAM still holds. With the lemma, we can safely use $e_i^{(m)}$ as a replacement for $x_i^{'}$ which is not under the influence of $x_m^{'}$. Collating multiple $e_i^{(m)}$ from the least squares regressions gives a [(p' – 1) × 1] random vector of error terms as $\mathbf{e}^{(m)}$:

$$\mathbf{e}^{(\mathrm{m})} = \left(e_1^{(m)}, \ldots, e_{m-1}^{(m)}, e_{m+1}^{(m)}, \ldots, e_{p'}^{(m)} \right)^{\mathrm{T}}$$

We redefine **x'** as $\mathbf{e}^{(m)}$ and exclude any variable subscript in K from U' (i.e., $U' \coloneqq U' \setminus K$). As long as the number of variable subscripts in U' is greater than one, DirectLiNGAM continues searching the first variable in **x'**. If the number of variable subscripts in U' is reduced to one, the remaining subscript would enter the tail of K. K is now complete as the causal order of observed variables. Table 2 gives an overview about the algorithm of DirectLiNGAM and Appendix 7 provides the technical details of DirectLiNGAM.

Taking the four-variable case as an example, we begin with an empty list K and a set of variable subscript U' = {a, b, c, d}. Let us say the M criterion indicates that x_{d} is the exogenous variable, with which we update K and U':

$$K\text{: } k(\mathrm{d});\ U' = \{\mathrm{a}, \mathrm{b}, \mathrm{c}\}$$

Since the causal order of x_{d} has been determined, it will no longer be included in the M criterion. The mission of M criterion is now changed to finding out the exogenous variable

between x_a, x_b and x_c (but not x_d). Assuming x_b is detected as the exogenous variable between x_a, x_b and x_c, we have K and U' updated the second time:

$$K\text{: } k(\text{d}) < k(\text{b}); \; U' = \{\text{a, c}\}$$

The pool of observed variables for M criterion is once again reduced to between x_a and x_c. Assuming x_c is detected as the exogenous variable between the two, we have K and U' updated the third time:

$$K\text{: } k(\text{d}) < k(\text{b}) < k(\text{c}); \; U' = \{\text{a}\}$$

Since one variable subscript remains in U', it is pointless to search for any exogenous variable with the M criterion. Simply we insert the remaining subscript "a" from U' to the tail of list K:

$$K\text{: } k(\text{d}) < k(\text{b}) < k(\text{c}) < k(\text{a}); \; U' = \emptyset \text{ (i.e., empty)}$$

Having a complete list of causal orders as K been obtained, DirectLiNGAM moves on to adaptive lasso regression which results in a matrix of regression coefficients as $\mathbf{\Gamma}$.

Table 2 *Algorithm Overview of DirectLiNGAM*

Input: Data matrix

Step 1: Set a list of observed variables as *U* and set an empty list as *K*. Repeat

Step 1.1: Determine the first observed variable of list *U*, based on the values of mutual information. First variable is defined as the one with no parent.

Step 1.2: Remove the effect of the first observed variable from the rest of observed variables on list *U* with least squares regressions.

Step 1.3: Drop the first observed variable from list *U* and add it to the end of list *K*.

Step 1.4: If list *U* has only one observed variable, break and add the remaining observed variable to list *K*. List *K* is complete as the causal order of observed variables.

Step 2: Estimate a matrix of regression coefficients with adaptive lasso regression.

Output: A matrix of regression coefficients between observed variables

2.2.3 ParceLiNGAM for Observed Variables

ParceLiNGAM (Tashiro et al., 2014) is an observed-variable causal discovery algorithm. While DirectLiNGAM is taking a top-down approach, ParceLiNGAM is a bottom-up approach in the way that it begins searching for the last variable, the second last variable, back to the first variable at the end. It had been first developed to discover the causal order of observed variables in LiNGAM with latent confounding variables (Hoyer et al., 2008). Yet this thesis will focus on the use of ParceLiNGAM in discovering the causal order of variable in standard LiNGAM without latent confounding variables involved.

As usual, suppose **x** as a ($p \times 1$) random vector of observed variables and $U = \{1, 2, \ldots, p\}$ as the respective set of observed variable subscripts; let *K* as an empty list of the causal order of observed variables (i.e., $K \coloneqq \emptyset$). We define **x**' as a ($p' \times 1$) random vector of

observed variables which is a duplicate of **x** (i.e., **x'** ≔ **x**), and define $U' = \{1, \ldots, p'\}$ as the corresponding set of observed variable subscripts. Consider a linear regression model that an observed variable, say x'_i, is regressed on the rest of the observed variables in **x'**:

$$x'_i = \sum_{j \in U';\, i \neq j}^{p'-1} \gamma_{ij}\, x'_j + e_i^{(-i)} \tag{2.14}$$

where γ_{ij} is the parameter that x'_i is regressed on the other observed variable x'_j; $e_i^{(-i)}$ is the error term. For the simplicity's sake, we let $\mathbf{x}'_{(-i)}$ be a $[(p' - 1) \times 1]$ vector of observed variables which is **x'** excluding x'_i. Tashiro and colleagues (2014) articulate that x'_i is the last observed variable if and only if $\mathbf{x}'_{(-i)}$ is independent of $e_i^{(-i)}$. As far as p' is greater than two, $\mathbf{x}'_{(-i)}$ is a vector and thus testing for the independence between $\mathbf{x}'_{(-i)}$ and $e_i^{(-i)}$ involves multiple pairwise independence tests, one x'_j from $\mathbf{x}'_{(-i)}$ against $e_i^{(-i)}$ at a time. An independence measure introduced earlier, mutual information, is not applicable in this case since mutual information in forms of one-dimensional differential entropies can only deal with one regressor and one error term.

To handle multiple regressors, ParceLiNGAM endorses Hilbert-Schmidt independence criterion (HSIC) with the Fisher's (1932) probability combination method (see Entner & Hoyer, 2011) as the independence measure. HSIC (Gretton et al., 2005, 2007) is a kernel method often used in machine learning that maps data from the (lower-dimensional) input space to the (higher-dimensional) reproducing kernel Hilbert space. The mathematical operation of HSIC is way too technical and beyond the scope of this thesis. We will solely touch on its surface below. Assume two variables, u and v. Let U be an input space of which u is an element (i.e., $u \in U$) and F be a reproducing kernel Hilbert space with respect to U. Let ϕ be a kernel function (or feature mapping) that maps u from U to F (ϕ: $U \rightarrow F$) such that $\phi(u)$ is an element of F (i.e., $\phi(u) \in F$). Let V be an input space of which v is an element (i.e.,

$v \in V$) and G be a reproducing kernel Hilbert space with respect to V. Let φ be a kernel function that maps v from V to G (φ: $V \rightarrow G$) such that $\varphi(v)$ is an element of G (i.e., $\varphi(v) \in G$). The cross-covariance operator between ϕ and φ (denoted by $\text{cov}(\phi, \varphi)$) is expressed as:

$$\text{cov}(\phi, \varphi) = E\{[\phi(u) - E(\phi(u))] \otimes [\varphi(v) - \text{E}(\varphi(v))]\} \quad (2.15)$$

where $\otimes$ denotes the Kronecker product. The square of the Hilbert-Schmidt norm of the cross-covariance operator is the definition of HSIC:

$$\text{HSIC}(u, v) = \left(\| \text{cov}(\phi, \varphi) \|_{\text{HS}} \right)^2 \quad (2.16)$$

where $P(u, v)$ denotes the joint probability distribution of variables u and v; "$\| . \|_{\text{HS}}$" is the operator of the Hilbert-Schmidt norm. The decision rule is that, if the value of $\text{HSIC}(u, v)$ is not statistically different from zero as evidenced by the p-value (i.e., $p_{\text{HSIC}}(u, v)$) greater than the alpha level of 0.05, it suggests that u is not dependent on v.

Given multiple $x_j^{'}$ in $\mathbf{x}_{(-i)}^{'}$, we undergo multiple pairwise independence tests using HSIC against $e_i^{(-i)}$, resulting in multiple p-values of HSIC. Since these probability values are in a uniform distribution, Fisher's method is a proper strategy to combine the probability values (Kost & McDermott, 2002). Fisher's method is to aggregate multiple p-values to a joint independence measure between $\mathbf{x}_{(-i)}^{'}$ and $e_i^{(-i)}$:

$$\chi^2_{\text{HSIC+Fisher}} \left(\mathbf{x}'_{(-i)}, e'^{(-i)}_i \right) = -2 \sum_{j=1}^{p'-1} \log \left[p_{\text{HSIC}} \left(x'_j, e'^{(-i)}_i \right) \right] \quad (2.17)$$

The statistic follows a chi-square distribution, with the degrees of freedom equal to [2 × (p' – 1)] and the p-value as $p_{\text{HSIC+Fisher}}$. The null hypothesis is that $e_i^{(-i)}$ is independent of every $x_j^{'}$ in $\mathbf{x}_{(-i)}^{'}$. According to Tashiro et al. (2014), for an observed variable x'_m that has the greatest-possible $p_{\text{HSIC+Fisher}}$, it implies $\mathbf{x}_{(-i)}^{'}$ and $e_m^{(-m)}$ have the lowest-possible likelihood to be

dependent, suggesting that $x_m^{'}$ qualifies for the last variable in **x'**. Mathematically, $x_m^{'}$ is defined as:

$$x'_m = \arg\max_{i \in U'} p_{\text{HSIC+Fisher}} \left(\mathbf{x}'_{(-i)}, e'^{(-i)}_i \right) \tag{2.18}$$

where "max" is the maximization function that selects a $x_I^{'}$ with the highest $p_{\text{HSIC+Fisher}}$ from **x'**. Once the $x_m^{'}$ is located, its subscript, *m*, enters the head of *K*. After that, we update **x**' as $\mathbf{x}_{(-m)}^{'}$ and exclude any subscript in *K* from *U'* (i.e., $U' \coloneqq U \setminus K$). Should the length of *U'* be greater than one, ParceLiNGAM continues searching the next (second) last variable in **x**'. If the length of *U'* equals one, the remaining subscript in *U'* is placed to the head of *K*. *K* is now complete as the causal order of variables. Table 3 shows the overview of how the algorithm of ParceLiNGAM operates and Appendix 8 provides the technical details.

Taking as an example the four-variable case (x_a, x_b, x_c and x_d) we repeatedly used, we start off with an empty list as *K* and a set of variable subscript *U'* = {a, b, c, d}. Let us say HSIC points out x_a is the last variable among the four. We update *K* and *U'* accordingly:

$$K\text{: } k(\text{a}); \; U' = \{\text{b, c, d}\}$$

We run the HSIC over again without taking x_a into account, since the position of the causal order of x_a has been identified. This time, HSIC indicates that x_c is the last variable between x_b, x_c and x_d. Thus, *K* and *U'* are revised to:

$$K\text{: } k(\text{c}) < k(\text{a}); \; U' = \{\text{b, d}\}$$

For the rest of the observed variables whose position in the causal order is yet to locate (x_b & x_d), we run the independence test whose results show x_b is the last variable. Now, *K* and *U'* become:

$$K\text{: } k(\text{b}) < k(\text{c}) < k(\text{a}); \; U' = \{\text{d}\}$$

ParceLiNGAM will cease to proceed due to one subscript remaining in *U'*. We complete *K* by adding that subscript to the head of *K*.

$$K\text{: } k(\text{d}) < k(\text{b}) < k(\text{c}) < k(\text{a}); \; U' = \emptyset$$

Table 3 *Algorithm Overview of ParceLiNGAM*

Input: Data matrix

Step 1: Set a list of observed variables as *U* and set an empty list as *K*. Repeat

Step 1.1: Determine the last observed variable of list *U*, based on Hilbert-Schmidt independence criterion test statistics. Last variable is defined as the one with no child.

Step 1.2: Drop the last observed variable from list *U* and add it to the top of list *K*.

Step 1.3: If list *U* has only one observed variable, break and add the remaining observed variable from list *U* to the top of list *K*. List *K* is complete as the causal order of observed variables.

Step 2: Estimate a matrix of regression coefficients with adaptive lasso regression.

Output: A matrix of regression coefficients between observed variables

Chapter 3: Present Study

3.1 Adapting LiNGAM to Latent Variable Models

Chapter 2 revisited the existing method for observed-variable causal discovery. The non-Gaussian approach is preferred to the Gaussian approach in the sense that the former approach generates one DAG as output, instead of multiple. This chapter is to develop a non-Gaussian causal discovery method for latent variables. In social sciences and psychology, many phenomena take places under the influence of implicit and unobserved entities. Latent-variable models are vital in encapsulating the abstraction of constructs which are not directly observable (Bollen, 2002). We extend LiNGAM and introduce the linear non-Gaussian acyclic model for latent variables (LiNGAM-LV) for latent-variable causal discovery, provided that the measurement model has been correctly specified by users (Anderson & Gerbing, 1988). As an extension of LiNGAM, LiNGAM-LV inherits from the four properties: linearity, non-Gaussianity, directedness and acyclicity. As with LiNGAM for observed variables, LiNGAM-LV observes CMC but not CFC. It means that, if a set of factors as X and the other set of factors as Y are *d*-separated by a set of factors as Z, they are independent conditional on Z. If X and Y are not *d*-separated, LiNGAM-LV will test for their independence without assuming that X and Y are dependent.

In the adaptation of LiNGAM to structural equation models, we have to keep in mind their differences which we summarize into two points. First, in LiNGAM-LV, the structural coefficient matrix as **B** can be permuted in strict lower triangularity and the relationships between latent variables are unidirectional. In structural equation models, the relationship between latent variables can be as either unidirectional or non-directional. Second, in LiNGAM-LV, residuals in the structural equations ($\boldsymbol{\zeta}$) and measurement errors in the measurement equations ($\boldsymbol{\varepsilon}$) are assumed in multivariate non-Gaussian distributions (Shimizu et al., 2009). In structural equation models, they are assumed in multivariate normal

distributions. We will propose and describe the three LiNGAM algorithms (ICA-LiNGAM, DirectLiNGAM & ParceLiNGAM) that operate in the context of latent-variable models in the following.

3.2 ICA-LiNGAM for Latent Variables

ICA-LiNGAM for latent variables (ICA-LiNGAM-LV) uses ICA with additive errors to find out the causal order of the latent variables. We will describe the algorithm of ICA-LiNGAM-LV in four steps: i) identifying LiNGAM-LV using ICA with additive errors, ii) permutating unmixing matrix to strict lower triangularity, iii) generating causal order, and iv) running regularized structural equation modelling.

3.2.1 Step 1: Identification of LiNGAM for Latent Variables using ICA with Additive Errors

From equation 1.2, the expression of the structural equations can be reduced to:

$$\boldsymbol{\eta} = (\mathbf{I} - \mathbf{B})^{-1}\boldsymbol{\zeta} \tag{3.1}$$

Putting the reduced form of the structural equations (equation 3.1) into the measurement equations (see equation 1.2), one obtains:

$$\mathbf{x} = \boldsymbol{\Lambda}(\mathbf{I} - \mathbf{B})^{-1}\boldsymbol{\zeta} + \boldsymbol{\varepsilon} \tag{3.2}$$

Standard ICA model does not take into account the error of measurement ($\boldsymbol{\varepsilon}$). We require a variant known as ICA model with additive errors (Shimizu et al., 2009), also known as the noisy ICA model. Suppose $\mathbf{s}$ is a ($h \times 1$) random vector of independent components. The definition of the ICA model with additive errors is:

$$\mathbf{x} = \mathbf{A}_{\text{ica}}\mathbf{s} + \mathbf{e} \tag{3.3}$$

where $\mathbf{A}_{\text{ica}}$ is a ($p \times h$) mixing matrix and $\mathbf{e}$ is a ($p \times 1$) random vector of additive errors. When p is large, ICA model with additive errors is computationally demanding. As a remedy, Shimizu and colleagues (2009) suggest a substitute which combines principal component analysis (PCA) with a standard ICA using a "FastICA" algorithm (Hyvärinen, 1999). If the

data is incomplete, iterative PCA (Josse & Husson, 2012; Josse & Husson, 2016) as an imputation method is performed prior to ICA with additive errors. Iterative PCA is compatible with non-zero correlations between variables and does not demand the assumption of multivariate Gaussian distribution (Dray & Josse, 2015).

To identify a LiNGAM-LV using ICA with additive errors, we have to set the number of independent components to be equal to the number of latent variables (i.e., $h = q$). It is apparent that equation 3.2 matches the definition of an ICA model with additive errors, with the relation about $\mathbf{A}_{\text{ica}}$:

$$\mathbf{A}_{\text{ica}} = \mathbf{\Lambda}(\mathbf{I} - \mathbf{B})^{-1} \tag{3.4}$$

Under the equality constraint that $h = q$, $\mathbf{A}_{\text{ica}}$ is a non-squared mixing matrix wherein the row is not on par with the column ($p \neq q$). The unmixing matrix as $\mathbf{W}_{\text{ica}}$ no longer defined as the inverse of $\mathbf{A}_{\text{ica}}$ but:

$$\mathbf{W}_{\text{ica}} = \left(\mathbf{\Lambda}^{\text{T}}\mathbf{A}_{\text{ica}}\right)^{-1}\left(\mathbf{\Lambda}^{\text{T}}\mathbf{\Lambda}\right) \tag{3.5}$$

By doing so, $\mathbf{W}_{\text{ica}}$ is corrected to be a ($h \times q$) squared matrix which is helpful in causal determination. As in the standard ICA model, the solution from ICA model with additive errors is vulnerable to permutation and scaling indeterminacies. A consequence brought by the two indeterminacies is that, of $\mathbf{W}_{\text{ica}}$, the sequence of latent variables in the row is arbitrary but the sequence in the column is known and follows $\boldsymbol{\eta}$. To illustrate, let us say a $\boldsymbol{\eta}$ be:

$$\boldsymbol{\eta} = \left(\eta_{\text{a}}, \eta_{\text{b}}, \eta_{\text{c}}, \eta_{\text{d}}\right)^{\text{T}}$$

and the known sequence of latent variables is {a, b, c, d}. Because of the permutation indeterminacy, the sequence of latent variables in the row of $\mathbf{W}_{\text{ica}}$ is unknown. It can be either {b, a, d, c}, {d, c, b, a} or others. It is very likely that the row of $\mathbf{W}_{\text{ica}}$ does not correspond to the column.

With the assistance of equation 2.7 (i.e., $\mathbf{W}_{\text{ica}} = \mathbf{P}\mathbf{D}\widetilde{\mathbf{W}}'_{\text{ica}}$), we can stave off two indeterminacies from $\mathbf{W}_{\text{ica}}$. The resultant $\widetilde{\mathbf{W}}'_{\text{ica}}$ is a $(q \times q)$ permutated unmixing matrix in which the sequence of latent variables in its row corresponds to the sequence of latent variables in its column. The ICA-estimated matrix about the connection strength for latent variables as $\mathbf{B}_{\text{ica}}$ can be obtained by:

$$\mathbf{B}_{\mathbf{ica}} = \mathbf{I} - \widetilde{\mathbf{W}}'_{\text{ica}} \tag{3.6}$$

Next step is to permute the $\mathbf{B}_{\text{ica}}$ in strict lower triangularity. That makes sure that the solution from ICA with additive errors meets acyclicity which is a property of LiNGAM.

3.2.2 Step 2: Permutation of Unmixing Matrix to Strict Lower Triangularity

The second step is to permute the $\mathbf{B}_{\text{ica}}$ from the first step to strict lower triangularity such that the permuted matrix suits the definition of $\mathbf{B}$ in LiNGAM for latent variables. Substitute $\mathbf{B}_{\text{ica}}$ into $\mathbf{\Gamma}_{\text{ica}}$ in equation 2.9, we have the mathematical expression of permuting $\mathbf{B}_{\text{ica}}$ to a strictly lower triangular matrix:

$$\widetilde{\mathbf{B}}_{\text{ica}} = \widetilde{\mathbf{P}}\mathbf{B}_{\text{ica}}\widetilde{\mathbf{P}}^{\text{T}} \tag{3.7}$$

where $\widetilde{\mathbf{P}}$ denotes a $(q \times q)$ permutation matrix. The resultant unmixing matrix $\widetilde{\mathbf{B}}_{\text{ica}}$ is a $(q \times q)$ unmixing matrix in strict lower triangularity. With $q > 2$, we take a method by Shimizu et al. (2011) that approximates the permutation of $\mathbf{B}_{\text{ica}}$ to alleviate the computation process. In the method, we replace $\mathbf{\Gamma}_{\text{ica}}$ with $\mathbf{B}_{\text{ica}}$ and replace p with q. After the permutation, the causal order of latent variables is known which can be presented using a list as K.

As stated earlier, ICA manages to separate two sources correctly but not more than two, as it struggles with the local maximum as opposed to the global maximum (Himberg et al., 2004). To reduce the odds of being stuck at suboptimal local maximum, ICA-LiNGAM-LV runs ICA with additive errors multiple times. At each time, ICA generates randomly a new set of starting values which could make ICA-LiNGAM-LV deducing K differently. With multiple Ks gathered, we compare and select the K with the greatest frequency as the final. If

a tie occurs, the algorithm generates additional *K*s, until the point at which the most frequent *K* can be located. All in all, one complete causal order of latent variables is resulted.

3.2.3 Step 3: Regularized Structural Equation Modelling

The causal order infers that latent variables at the earlier positions are the ancestors of those at the later positions. Yet, the causal order does not tell how directed edges connect between latent variables. That is, just because a latent variable η_i is ahead of the other latent variable η_j in the causal order (i.e., $k(i) < k(j)$) does not necessarily implies that η_i extends a pathway to η_j. To find out the causal relationship between latent variables under the guidance of their causal order obtained from Step 2, LiNGAM-LV algorithms undergoes regularized structural equation modelling (RegSEM).

RegSEM (Jacobucci et al., 2016) is the integration of a standard structural equation modelling which involves a statistical technique called regularization. Regularization (shrinkage or penalization) is a method that makes use of some cost functions to incrementally penalize the size of some model parameters in an iterative manner, leading to a relatively more parsimonious models to ease result interpretation. However, overly penalizing the size of model parameters can cause model misfit. RegSEM aims at striving a balance between model parsimoniousness and model fit with information criteria (e.g., BIC) and fit indices (e.g., RMSEA) to monitor the change in model parsimoniousness and model fit. The major advantage of RegSEM over a standard SEM is the increased flexibility in lowering model complexity and simultaneously avoid poorly fitted models (Jacobucci et al., 2016).

The cost function of RegSEM (denoted by F_{RegSEM}) is:

$$F_{\text{RegSEM}} = F_{\text{ML}} + \lambda P(.) \tag{3.8}$$

where F_{ML} denotes the fit function of maximum likelihood estimation, λ denotes the regularization parameter (or penalty) ranging from zero to positive infinity, and "$P(.)$" denotes a function of aggregating model parameters which changes with various

regularization methods, such as lasso (Tibshirani, 1996) and adaptive lasso (Zou, 2006). The definition of F_{ML} is:

$$F_{\mathrm{ML}} = \log|\Sigma(\boldsymbol{\theta})| + \mathrm{tr}\left(\mathbf{S} \times \Sigma^{-1}(\boldsymbol{\theta})\right) - \log|\mathbf{S}| - p \tag{3.9}$$

where $\boldsymbol{\theta}$ is a set of model parameters, $\Sigma(\boldsymbol{\theta})$ is the model-implied covariance matrix, and $\mathbf{S}$ is the sample covariance matrix. When λ is set to zero, $F_{\mathrm{RegSEM}} = F_{\mathrm{ML}}$. RegSEM is implementable in a R package called "regsem" (Jacobucci et al., 2021). Due to its computational complexity, standard errors of parameters are not available (Liang & Jacobucci, 2020).

With the complete causal order of latent variables as K, ICA-LiNGAM-LV harnesses RegSEM to detect causal relationships between the latent variables. In the following we will demonstrate how ICA-LiNGAM-LV operates with RegSEM. Suppose $\mathbf{x} = (x_{\mathrm{a1}}, x_{\mathrm{a2}}, x_{\mathrm{a3}}, x_{\mathrm{b1}}, x_{\mathrm{b2}}, x_{\mathrm{b3}}, x_{\mathrm{c1}}, x_{\mathrm{c2}}, x_{\mathrm{c3}}, x_{\mathrm{d1}}, x_{\mathrm{d2}}, x_{\mathrm{d3}})^{\mathrm{T}}$ is a vector of 12 observed variables and $\boldsymbol{\eta} = (\eta_{\mathrm{a}}, \eta_{\mathrm{b}}, \eta_{\mathrm{c}}, \eta_{\mathrm{d}})^{\mathrm{T}}$ is a vector of four latent variables; The measurement model is a four-factor model wherein η_{a} is loaded on three observed indicators, x_{a1}, x_{a2} and x_{a3}; η_{b} is loaded on x_{b1}, x_{b2} and x_{b3}; η_{c} is loaded on x_{c1}, x_{c2} and x_{c3}; η_{d} is loaded on x_{d1}, x_{d2} and x_{d3}. In the path model, η_{d} is an exogenous latent variable which has a direct effect on η_{d} and on η_{c}; η_{b} has a direct effect on η_{c}; and η_{c} has a direct effect on η_{a} (The path diagram is the same as shown in Figure 7a, except for the observed variables are replaced with the latent variables). The causal order is thereby K: $k(\mathrm{d}) < k(\mathrm{b}) < k(\mathrm{c}) < k(\mathrm{a})$. We substantiate $\mathbf{B} = \{b_{ij}\}$ as a (4 × 4) null matrix for all i, j = (a, b, c, d). As latent variables cannot have a direct effect upon themselves, we set the diagonal elements in $\mathbf{B}$ to zeros:

$$\mathbf{B} = \begin{pmatrix} b_{\mathrm{aa}} & b_{\mathrm{ab}} & b_{\mathrm{ac}} & b_{\mathrm{ad}} \\ b_{\mathrm{ba}} & b_{\mathrm{bb}} & b_{\mathrm{bc}} & b_{\mathrm{bd}} \\ b_{\mathrm{ca}} & b_{\mathrm{cb}} & b_{\mathrm{cc}} & b_{\mathrm{cd}} \\ b_{\mathrm{da}} & b_{\mathrm{db}} & b_{\mathrm{dc}} & b_{\mathrm{dd}} \end{pmatrix} = \begin{pmatrix} 0 & b_{\mathrm{ab}} & b_{\mathrm{ac}} & b_{\mathrm{ad}} \\ b_{\mathrm{ba}} & 0 & b_{\mathrm{bc}} & b_{\mathrm{bd}} \\ b_{\mathrm{ca}} & b_{\mathrm{cb}} & 0 & b_{\mathrm{cd}} \\ b_{\mathrm{da}} & b_{\mathrm{db}} & b_{\mathrm{dc}} & 0 \end{pmatrix}$$

η_d is identified as the first variable amongst the four. It has no reason to regress η_d on any other variables and thereby we set all elements in the fourth row to zeros, after which **B** looks like:

$$\mathbf{B} = \begin{pmatrix} 0 & b_{ab} & b_{ac} & b_{ad} \\ b_{ba} & 0 & b_{bc} & b_{bd} \\ b_{ca} & b_{cb} & 0 & b_{cd} \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Next, since $k(d) < k(b)$, we are to test the presence of the causal effect of η_d on η_b, we construct a latent-variable model where the measurement model has two common factors, η_d and η_b, along with the corresponding observed indicators (i.e., x_{d1}, x_{d2}, x_{d3}, x_{b1}, x_{b2} & x_{b3}). In the path model, η_d has a direct effect on η_b and the estimate of the structural parameter by RegSEM is denoted by $\bar{b}_{bd}$ which may or may not be penalized to zero. Since neither η_a nor η_c are in the model, we set b_{da} and b_{dc} to zeros. **B** is updated to:

$$\mathbf{B} = \begin{pmatrix} 0 & b_{ab} & b_{ac} & b_{ad} \\ 0 & 0 & 0 & \bar{b}_{bd} \\ b_{ca} & b_{cb} & 0 & b_{cd} \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Moving on to the third variable, η_c, since $k(d) < k(b) < k(c)$, we are intrigued in whether η_c is regressed on η_b and η_d. This time, we construct a new latent-variable model. The measurement model has three common factors (η_c, η_d & η_b) and nine corresponding observed indicators (x_{c1}, x_{c2}, x_{c3}, x_{d1}, x_{d2}, x_{d3}, x_{b1}, x_{b2} & x_{b3}). In the path model, η_d and η_b each has a direct effect on η_c, whose structural coefficients estimated by RegSEM are denoted by $\bar{b}_{cd}$ and $\bar{b}_{cb}$, respectively. No pathway is specified between η_d and η_b here. Since η_a is absent from the model, we set b_{ca} to zero. Thus **B** becomes:

$$\mathbf{B} = \begin{pmatrix} 0 & b_{ab} & b_{ac} & b_{ad} \\ 0 & 0 & 0 & \bar{b}_{bd} \\ 0 & \bar{b}_{cb} & 0 & \bar{b}_{cd} \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

η_a as the last variable is regressed on η_b, η_c and η_d. Once again, we construct a new latent-variable model. This time, the measurement model has all four common factors and all 12

observed indicators where the four factors are loaded onto their corresponding observed indicators. In the path model, η_d, η_b and η_c each has a direct effect on η_a, whose structural coefficients estimated by RegSEM are denoted $\bar{b}_{ad}$, $\bar{b}_{ab}$ and $\bar{b}_{ac}$, respectively. **B** now looks like:

$$\mathbf{B} = \begin{pmatrix} 0 & \bar{b}_{ab} & \bar{b}_{ac} & \bar{b}_{ad} \\ 0 & 0 & 0 & \bar{b}_{bd} \\ 0 & \bar{b}_{cb} & 0 & \bar{b}_{cd} \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

As far, we have performed three rounds of RegSEM. In each round, only one latent variable was set as an endogenous latent variable and RegSEM selectively penalizes those structural coefficients in the same row of **B**. To allow RegSEM for simultaneously penalizing six structural coefficients present in **B**, we undergo one more RegSEM. In the specification of the latent-variable model for the final round of RegSEM, the measurement model is a four-factor model that contains all four common factors and all 12 observed indicators, with the four factors being loaded onto the corresponding observed indicators. The specification of the path model follow what **B** describes. That is, η_d extends one pathway to η_b and one pathway to η_c; η_b extend a pathway to η_c; η_c extends a pathway to η_d. No covariance is allowed between latent variables here. In doing so, we allow RegSEM to penalize all six structural coefficients ($\bar{b}_{bd}$, $\bar{b}_{cd}$, $\bar{b}_{cb}$, $\bar{b}_{ad}$, $\bar{b}_{ab}$ & $\bar{b}_{ac}$) in the same model. The finalized **B** is the output of ICA-LiNGAM-LV which is in strict lower triangular (Bollen, 1989) for one can easily permute the finalized **B** to:

$$\mathbf{B} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ \bar{b}_{bd} & 0 & 0 & 0 \\ \bar{b}_{cd} & \bar{b}_{cb} & 0 & 0 \\ \bar{b}_{ad} & \bar{b}_{ab} & \bar{b}_{ac} & 0 \end{pmatrix}$$

The overview about the algorithm ICA-LiNGAM-LV is shown in Table 4 and the technical detail is present in Appendix 9.

Table 4 *Algorithm Overview of ICA-LiNGAM-LV*

Input: Data matrix and factor loading matrix from factor-analysis model

Step 1: Repeat *N* times to accumulate a total of *N* causal orders of latent variables.

Step 1.1: Estimate an unmixing matrix with independent component analysis with additive errors, where the number of independent components is set to the number of latent variables.

Step 1.2: Permute the unmixing matrix to the one filled with non-zero diagonal elements.

Step 1.3: Standardize the permuted unmixing matrix.

Step 1.4: Obtain the causal order of latent variables.

Step 2: Select a causal order of all with the highest frequency. If a tie happens, repeat Step 1.

Step 3: Estimate a matrix of structural coefficients between latent variables with regularized structural equation modelling.

Output: A matrix of structural coefficients between latent variables

3.3 DirectLiNGAM for Latent Variables

DirectLiNGAM for latent variables (DirectLiNGAM-LV) is an estimation method that identifies a LiNGAM for latent variables. DirectLiNGAM-LV relies on the differences in mutual information as an independence measure to determine the causal order of latent variables from top down. Once the causal order of the latent variables is complete, DirectLiNGAM-LV ends with regularized structural equation modelling to discover the structural coefficient matrix.

3.3.1 Factor Score Approximation

The original DirectLiNGAM makes use of mutual information as an independence measure to sort out the causal order of observed variables. The computation of mutual

information requires the true scores on the observed variables. Yet we encountered a challenge when applying DirectLiNGAM to latent-variable models since the scores on latent variables are not observable.

As a solution, we use factor scores to approximate the true scores on latent variables. Factor scores are the regression estimates of the true scores on the latent variables which can be calculated through various approaches. This thesis will look at three approaches which are Thurstone (1937), Bartlett (1937) and ten Berge et al. (1976), in chronological order. Let $\mathbf{f} = \left(f_1, \ldots, f_q\right)^{\mathrm{T}}$ as a ($q \times 1$) random vector of factor scores on the latent variables (and $\boldsymbol{\eta} = \left(\eta_1, \ldots, \eta_q\right)^{\mathrm{T}}$ is the vector of the true scores on the latent variables). $\mathbf{f}$ approximated by Thurstone / Bartlett / ten Berge et al. are dubbed as $\mathbf{f}^{(1)}$ / $\mathbf{f}^{(2)}$ / $\mathbf{f}^{(3)}$. The respective expressions are:

$$\mathbf{f}^{(1)} = (\boldsymbol{\Lambda}\boldsymbol{\Phi})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\mathbf{x} \tag{3.10}$$

$$\mathbf{f}^{(2)} = \left[\boldsymbol{\Lambda}\left(\boldsymbol{\Lambda}^{\mathrm{T}}\mathbf{U}^{-2}\boldsymbol{\Lambda}\right)^{-1}\right]^{\mathrm{T}}\boldsymbol{\Sigma}\,\mathbf{x} \tag{3.11}$$

$$\mathbf{f}^{(3)} = \left(\mathbf{C}\boldsymbol{\Phi}^{\frac{1}{2}}\right)^{\mathrm{T}}\boldsymbol{\Sigma}\,\mathbf{x} \tag{3.12}$$

where $\boldsymbol{\Sigma}$ is a ($p \times p$) covariance-variance matrix of observed variables, and

$$\mathbf{U}^2 = 1 - \mathrm{diag}\left(\boldsymbol{\Lambda}\boldsymbol{\Phi}\boldsymbol{\Lambda}^{\mathrm{T}}\right) \tag{3.13}$$

$$\mathbf{U}^{-2} = \mathrm{diag}\left(\frac{1}{\mathbf{U}^2}\right) = \mathrm{diag}\left(\frac{1}{1 - \mathrm{diag}\left(\boldsymbol{\Lambda}\boldsymbol{\Phi}\boldsymbol{\Lambda}^{\mathrm{T}}\right)}\right) \tag{3.14}$$

$$\mathbf{C} = \boldsymbol{\Sigma}^{-\frac{1}{2}}\mathbf{L}\left(\mathbf{L}^{\mathrm{T}}\boldsymbol{\Sigma}\mathbf{L}\right)^{-\frac{1}{2}} \tag{3.15}$$

$$\mathbf{L} = \boldsymbol{\Lambda}\boldsymbol{\Phi}^{\frac{1}{2}} \tag{3.16}$$

Grice (2001) reviews various factor-score approximation approaches and concludes that the three approaches each has their own strength. Neither of the three wholly outperform the

others but are equally promising. Therefore, the three approaches will be used to approximate factor scores in DirectLiNGAM-LV.

The usage of factor scores comes with caution. Factor-score indeterminacy is the fact that the scores on the common factors and the scores on the unique factors (or measurement errors) cannot be uniquely determined solely by the observed variables (McDonald & Mulaik, 1979). Although factor analysis (e.g., CFA) offers a solution to the measurement model, it is not sufficient to uniquely identify the latent variables as well as the measurement errors (Mulaik, 2005). In theory, one can construct an infinite set of scores on the latent variables that behave just like what the solution of the factor-analysis model (e.g., $\mathbf{\Lambda}$, $\mathbf{\Phi}$, $\boldsymbol{\varepsilon}$) describes. Factor-score regression has been under criticism (Estabrook & Neale, 2013) for the factor-score estimated structural parameters can be biased (e.g., Hayes & Usami, 2020). Therefore, factor scores are not involved in estimating the causal relationships between latent variables. They will be used to determine the causal order of latent variables only in DirectLiNGAM-LV.

3.3.2 Determine the First Latent Variable in Causal Order

Define $\mathbf{f}' = \left(f_1', \ldots, f_q'\right)^{\mathrm{T}}$ as a ($q' \times 1$) random vector of factor scores which is a duplicate of **f** (i.e., $\mathbf{f}' \coloneqq \mathbf{f}$). As will be discussed in the following, the number of factor scores in **f'** will decrease in the iterative process when more latent variables have their positions being identified in the causal order of the latent variables. To preserve **f,** we duplicate **f** and name it as **f'**. From now on, any changes will be applied to the duplicated **f'**.

Between two factor scores (f_i' and f_j') in **f'**, we wish to locate which is the cause. By running factor-score regressions between the two, we obtain an error term $e_i^{(j)}$ corresponding to f_i' and an error term $e_j^{(i)}$ corresponding to f_j'. From Shimizu et al. (2011), it is proved that a f_i' is said to be exogenous if the f_i' is independent of $e_j^{(i)}$. They further suggest that, if the independence

between f_i' and $e_j^{(i)}$ is greater than that of f_j' and $e_i^{(j)}$, f_i' is regarded as the cause between the two As in DirectLiNGAM, the independence is quantified with mutual information (MI; Hyvärinen, 1998). MI between f_i' and $e_j^{(i)}$ is denoted by $\mathrm{MI}\left(f_i', e_j^{(i)}\right)$. The difference between $\mathrm{MI}\left(f_i', e_j^{(i)}\right)$ and $\mathrm{MI}\left(f_j', e_i^{(j)}\right)$ is indicative of which is the cause between f_i' and f_j':

$$\Delta\mathrm{MI} = \mathrm{MI}\left(f_j', e_i^{(j)}\right) - \mathrm{MI}\left(f_i', e_j^{(i)}\right) = H\left(f_j'\right) + H\left(e_i^{(j)}\right) - H\left(f_i'\right) - H\left(e_j^{(i)}\right) \tag{3.17}$$

where *H(u)* denotes the one-dimensional differential entropy of variable *u*. The interpretation of ΔMI is as follows. If ΔMI > 0, it infers that the level of independence between f_i' and $e_j^{(i)}$ is greater than that that between f_j' and $e_i^{(j)}$, leading to a conclusion that f_i' is the cause. If ΔMI < 0, we draw an opposite conclusion that f_j' is the cause. In short, ΔMI is suggestive of which is the cause between two factor scores.

To search the first variable from all, DirectLiNGAM-LV conducts pairwise comparison of a variable f_i' against every other variable f_j' in **f'**, resulting in multiple ΔMIs. *M* criterion for a f_i' that collates multiple ΔMIs can be expressed as (Hyvärinen & Smith, 2013):

$$\begin{aligned} M(f_i'; U'\backslash i) &= \sum_{j\neq i, j=1}^{q'} \min(0, \Delta\mathrm{MI})^2 \\ &= \sum_{j\neq i, j=1}^{q'} \min\left(0, \mathrm{MI}\left(f_j', e_i^{(j)}\right) - \mathrm{MI}\left(f_i', e_j^{(i)}\right)\right)^2 \end{aligned} \tag{3.18}$$

where *U'* is a set of *q'* latent-variable subscripts from **f'**; *U'/i* represents *q'* – 1 subscripts in *U'* excluding *i*. "min" is the minimization function. The possible range of *M* criterion is from zero to positive infinite. If the *M* criterion for f_i' becomes positively infinite, ΔMIs are always larger than zero and the level of independence between f_i' and $e_j^{(i)}$ is consistently greater than the level of independence between f_j' and $e_i^{(j)}$, meaning f_i' is always the cause. It concludes that

the corresponding latent variable η_i is likely an exogenous latent variable. If the *M* criterion for $f_i^{'}$ comes close to zero, it implies that ΔMIs are not always at positive values and $f_i^{'}$ is less likely to be the cause of $f_i^{'}$ between the pair, leading to a conclusion that the corresponding latent variable η_i is less likely an exogenous latent variable.

3.3.3 Remove the Effect of the First Latent Variable from the Rest

From **f'**, let us say a factor score is found to have the greatest possible value of *M* criterion which we denote as $f_m^{'}$. $f_m^{'}$ is viewed as the exogenous variable in **f'**. We substantiate an empty list as *K* to keep track of the causal order of the latent variables in **η** (i.e., $K \coloneqq \emptyset$). The subscript of $f_m^{'}$, which is "*m*", enters the tail of *K*, indicating that η_m holds the first position in the causal order. Prior to searching for the second latent variable in the causal order, it is necessary to remove the undesirable effect of $f_m^{'}$ from all other factor scores in **f'**. To do so, we run least square regressions of $f_m^{'}$ on every other $f_i^{'}$ in **f'** ($i = 1, \ldots, q'$ for all $i \neq m$) in which we retrieve the error term as $e_i^{(m)}$:

$$e_i^{(m)} = f_i^{'} - \bar{\gamma}_{im} f_m^{'}$$

where $\bar{\gamma}_{im}$ is the estimate of the regression coefficient of $f_i^{'}$ on $f_m^{'}$. Shimizu et al. (2011) in their Lemma 2 state that $e_i^{(m)}$ captures the essence of $f_i^{'}$ without being influenced by $f_m^{'}$. $e_i^{(m)}$ is a preferable score representing $f_i^{'}$, based on which one continues to search for the second latent variable coming after η_m in the causal order. Collating a total of $q' - 1$ error terms from the least square regressions constitutes a $q' - 1$ random vector as $\mathbf{e}^{(m)}$:

$$\mathbf{e}^{(m)} = \left(e_1^{(m)}, \ldots, \mathrm{e}_{m-1}^{(m)}, \mathrm{e}_{m+1}^{(m)}, \ldots, \mathrm{e}_{q'}^{(m)} \right)^{\mathrm{T}}$$

f' is redefined as $\mathbf{e}^{(m)}$ and we exclude the latent variable subscript "*m*" from *U'* (i.e., $U' \coloneqq U' \setminus m$). So the number of subscripts in *U'* is now reduced by 1.

As far as the number of subscripts in U' is greater than 1, DirectLiNGAM-LV continues searching the exogenous latent variable with the redefined **f'**. If the number of subscripts in U' equates to 1, the remaining latent variable subscript goes to the tail of K. K is completed that represents the causal order of the latent variables in **η**. With the completed K, DirectLiNGAM-LV ends with running a series of RegSEM and discovers the causal relationship between latent variables which can be presented with a structural coefficient matrix in strict lower triangular. Table 5 is the overview of DirectLiNGAM-LV and Appendix 10 shows the technical details.

To illustrate how DirectLiNGAM-LV manages to determine the causal order of latent variables, imagine we have a $\boldsymbol{\eta} = (\eta_a, \eta_b, \eta_c, \eta_d)^T$ which is approximated by $\mathbf{f} = (f_a, f_b, f_c, f_d)^T$. Define $\mathbf{f}' \coloneqq \mathbf{f}$, with $U' = \{a, b, c, d\}$ as a set of the latent variable subscripts. We start off with an empty list as K. Amongst the four factor scores in **f'**, it has shown that:

$$\max\left(M(\dot{f}_a; U'\backslash a), M(\dot{f}_b; U'\backslash b), M(\dot{f}_c; U'\backslash c), M(\dot{f}_d; U'\backslash d)\right) = M(\dot{f}_d; U'\backslash d)$$

an indication that η_d is the exogenous latent variable. With such information, we update K and U' by adding the latent variable subscript "d" to the end of K and removing "d" from U':

$$K: k(d);\ U' = \{a, b, c\}$$

With the position of η_d in the causal order being determined, we conduct least square regressions of η_a, of η_b and of η_c on η_d. From each of the three regressions, we obtain three error terms, $e_a^{(d)}$, $e_b^{(d)}$ and $e_c^{(d)}$, respectively, and collate them to form a (3 × 1) vector of error terms as $\dot{\mathbf{f}} = (e_a^{(d)}, e_b^{(d)}, e_c^{(d)})^T$ which is isolated from the effect of $\dot{f}_d$. In the newly defined **f'**, we have:

$$\max\left(M(e_a^{(d)}; U'\backslash a), M(e_b^{(d)}; U'\backslash b), M(e_c^{(d)}; U'\backslash c)\right) = M(e_b^{(d)}; U'\backslash b)$$

a sign that η_b is the second latent variable in the causal order. Thus, we can update K and U' to:

$$K\text{: } k(\text{d}) < k(\text{b});\ U' = \{\text{a, c}\}$$

Then, we run least squares regressions of $e_{\text{a}}^{(\text{d})}$ and of $e_{\text{c}}^{(\text{d})}$ on $e_{\text{b}}^{(\text{d})}$ and obtain two error terms, $e_{\text{a}}^{(\text{d,b})}$ and $e_{\text{c}}^{(\text{d,b})}$. $\mathbf{f}'$ is once again redefined to $\mathbf{f}' = \left(e_{\text{a}}^{(\text{d,b})}, e_{\text{c}}^{(\text{d,b})}\right)^{\text{T}}$ which is not under the influence of $e_{\text{b}}^{(\text{d})}$. In $\mathbf{f}'$ that has been redefined the second time, the M criterion of $e_{\text{c}}^{(\text{d,b})}$ is found to be the greatest and thus η_{c} is the third latent variable in the causal order, with which we renew update K and U' to:

$$K\text{: } k(\text{d}) < k(\text{b}) < k(\text{c});\ U' = \{\text{a}\}$$

Since we have only one subscript remaining in U', that subscript “a” will insert to the tail of K, leaving U' in barrel.

$$K\text{: } k(\text{d}) < k(\text{b}) < k(\text{c}) < k(\text{a});\ U' = \emptyset$$

It results in K as a causal order of the four latent variables.

Table 5 *Algorithm Overview of DirectLiNGAM-LV*

Input: Data matrix, factor loading matrix and factor covariance matrix from factor-analysis model

Step 1: Set a list of latent variables as *U* and set an empty list as *K*. Compute the factor scores on latent variables and replace the true scores on latent variables with their factor scores. Repeat

Step 1.1: Determine the first latent variable of list *U*, based on the mutual information. First variable is defined as the one with no parent.

Step 1.2: Remove the effect of the first latent variable from the rest of latent variables on list *U* with least squares regressions.

Step 1.3: Drop the first latent variable from list *U* and add it to the end of list *K*.

Step 1.4: If list *U* has only one latent variable, break and add the remaining latent variable to the end of list *K*. List *K* is complete as the causal order of latent variables.

Step 2: Discard the factor scores. Estimate a matrix of structural coefficients between latent variables with regularized structural equation modelling.

Output: A matrix of structural coefficients between latent variables

3.4 ParceLiNGAM for Latent Variables

ParceLiNGAM for latent variables (ParceLiNGAM-LV) is the third algorithm this thesis introduces that identifies LiNGAM for latent variables. ParceLiNGAM-LV uses HSIC (Gretton et al., 2005; Gretton et al., 2007) with the Fisher (1932)'s probability combination method as an independence measure to determine the causal order of latent variables from bottom up. As with DirectLiNGAM-LV, ParceLiNGAM-LV also requires the factor scores for the independence measure.

Suppose $\boldsymbol{\eta} = \left(\eta_1 \ldots, \eta_q\right)^{\mathrm{T}}$ is a ($q \times 1$) vector of latent variables and $\mathbf{f} = \left(f_1, \ldots, f_{\mathrm{q}}\right)^{\mathrm{T}}$ is a vector of the corresponding factor scores, with $U = \{1, \ldots, q\}$ as a set of the latent-variable subscripts. Let K be an empty list. To preserve **f** in the repeated iteration, we duplicate **f** as **f'**, with $U' = \{1, \ldots, q'\}$. In **f'**, the relationship of a $f_i^{'}$ with other factor scores can be described with a linear regression model:

$$f_i' = \sum_{j \in U_f \setminus K, i \neq j}^{q'-1} \gamma_{ij} f_j' + e_i^{(-i)} \tag{3.19}$$

for all $i \in U'$, where γ_{ij} is the regression coefficient of $f_i^{'}$ on $f_j^{'}$ for all $j \in U'$, $i \neq j$; and $e_i^{(-i)}$ is the error term. In equation 3.19, we can express the regressors with a vector of $\mathbf{f}_{(-i)}^{'} = \left(f_1^{'}, \ldots, f_{i-1}^{'}, f_{i+1}^{'}, \ldots, f_{q'}^{'}\right)^{\mathrm{T}}$. Tashiro et al. (2014) prove that a variable is considered the last variable in the causal order if and only if its regressors are independent of the error term. Referring to equation 3.19, we make an approximation that a latent variable η_i is viewed as the last latent variable if and only the regressors of its factor score $f_i^{'}$ (denoted by $\mathbf{f}_{(-i)}^{'}$) are independent of $e_i^{(-i)}$. Since the number of the regressors can be more than one, ParceLiNGAM-LV runs multiple independence tests of HSIC (Gretton et al., 2005, 2007). At each time, it tests for the independence between one of the $f_j^{'}$ in $\mathbf{f}_{(-i)}^{'}$ and $e_i^{(-i)}$. The HSIC between a $f_j^{'}$ and $e_i^{(-i)}$ is denoted by $\mathrm{HSIC}\left(f_j^{'}, e_i^{(-i)}\right)$ whose p-value is $p_{\mathrm{HSIC}}(f_j^{'}, e_i^{(-i)})$. As long as we have more than one $f_j^{'}$ in $\mathbf{f}_{(-i)}^{'}$, ParceLiNGAM-LV uses Fisher's (1932) probability combination method to gather the p-values of HSIC. HSIC in association with Fisher's method forms a joint independence measure which is expressed as:

$$\chi^2_{\mathrm{HSIC+Fisher}}\left(\mathbf{f}_{(-i)}', e_i^{(-i)}\right) = -2 \sum_{j \in U_f; i \neq j}^{q'-1} \log[p_{\mathrm{HSIC}}(f_j', e_i^{(-i)})] \tag{3.20}$$

The HSIC test statistic with Fisher's method (denoted by $x^2_{\text{HSIC+Fisher}}$) is in a chi-square distribution, with the degrees of freedom equal to [2 × (q' – 1)]. The respective p-value is presented as $p_{\text{HSIC+Fisher}}$. In its null hypothesis, $e_i^{(-i)}$ is independent of every factor score in $\mathbf{f}'_{(-i)}$. Tashiro et al. (2014) introduce a way to interpret $p_{\text{HSIC+Fisher}}$. For a variable with the greatest-possible $p_{\text{HSIC+Fisher}}$, it suggests the least likelihood that its set of regressors are dependent on the error term. The concerned variable is said to be the last variable in the causal order. In our case, if a factor score f'_m has the greatest $p_{\text{HSIC+Fisher}}$ in $\mathbf{f}'$, it means $\mathbf{f}'_{(-m)}$ is of the least probability to be dependent on $e_m^{(-m)}$, yielding a conclusion that the corresponding η_m is the last latent variable in the causal order. f'_m can be located through:

$$f'_m = \arg\max_{i \in U_f \setminus K} p_{\text{HSIC+Fisher}}\left(\mathbf{f}'_{(-i)}, e_i^{(-i)}\right) \tag{3.21}$$

With f'_m being identified, the subscript "m" enters the head of K. Then, we update $\mathbf{f}'$ to be $\mathbf{f}'_{(-m)}$ (i.e., $\mathbf{f}' \coloneqq \mathbf{f}'_{(-m)}$) and remove the subscript "m" from U'. If the number of latent-variable subscripts in U' exceeds one, ParceLiNGAM-LV continues searching the second last variable in $\mathbf{f}'$. Should the number of latent-variable subscripts in U' equal one, the remaining subscript in U' goes to the head of K. Subsequently, K is complete as the causal order of the latent variables in $\boldsymbol{\eta}$. Table 6 gives the overview of ParceLiNGAM-LV and Appendix 11 shows the technicality.

To illustrate how ParceLiNGAM-LV is in search for a causal order of latent variables, conceive a (4 × 1) random vector of latent variables as $\boldsymbol{\eta} = \left(\eta_{\text{a}}, \eta_{\text{b}}, \eta_{\text{c}}, \eta_{\text{d}}\right)^{\text{T}}$, approximated by a vector of factor scores as $\mathbf{f} = \left(f_{\text{a}}, f_{\text{b}}, f_{\text{c}}, f_{\text{d}}\right)^{\text{T}}$, with a set of the latent-variable subscripts as U = {a, b, c, d}. We substantiate an empty list as K. Let us say it has indicated that f_{a} owns the greatest $p_{\text{HSIC+Fisher}}$ amongst the four factor scores in $\mathbf{f}$, meaning η_{a} is the last latent variable in the causal order. The subscript "a" enters the head of K from U:

$$K\text{: } k(\text{a}); \; U' = \{\text{b, c, d}\}$$

As f_{a} is out of consideration, we perform the independence test again for f_{b}, f_{c} and f_{d}, the results of which have shown that η_{c} is the last latent variable among the three and is the second last latent variable of all. We update K and U accordingly:

$$K\text{: } k(\text{c}) < k(\text{a}); \; U' = \{\text{b, d}\}$$

Between f_{b} and f_{d}, the HSIC in association with Fisher's method shows that f_{b} possesses the greatest $p_{\text{HSIC+Fisher}}$. η_{d} is viewed as the third last variable of all. We revise K and U to:

$$K\text{: } k(\text{b}) < k(\text{c}) < k(\text{a}); \; U' = \{\text{d}\}$$

We complete the causal order of the four latent variables by adding the latent-variable subscript remaining in U' to the head of K.

$$K\text{: } k(\text{d}) < k(\text{b}) < k(\text{c}) < k(\text{a}); \; U' = \emptyset$$

Table 6 *Algorithm Overview of ParceLiNGAM-LV*

Input: Data matrix, factor loading matrix and factor covariance matrix from factor-analysis model

Step 1: Set a list of latent variables as *U* and set an empty list as *K*. Compute the factor scores on latent variables and replace the true scores on latent variables with their factor scores. Repeat

Step 1.1: Determine the last latent variable of list *U*, based on Hilbert-Schmidt independence criterion test statistics. Last variable is defined as the one with no child.

Step 1.2: Drop the last latent variable from list *U* and add it to the top of list *K*.

Step 1.3: If list *U* has only one latent variable, break and add the remaining latent variable to the top of list *K*. List *K* is complete as the causal order of latent variables.

Step 2: Discard the factor scores. Estimate a matrix of structural coefficients between latent variables with regularized structural equation modelling.

Output: A matrix of structural coefficients between latent variables

Chapter 4: Method

4.1 Study Design

The simulation study covers three aspects which are data missingness, data non-normality and sample size. Parameters were estimated using maximum likelihood estimation with robust standard errors ("MLR") method as previous studies found that MLR method is robust to the violation of the multivariate normality assumption (Yuan & Bentler, 2000; Yuan, 2005) with small and medium sample sizes.

4.1.1 Population Model

The choice of population models that are conceptually meaningful and commonly countered in practice warrants the external validity of a simulation study (Gerbing & Anderson, 1993). The current study considered three. Population Model 1 (PM1) was a causal chain model commonly used in simulation studies (e.g., Chen, 2008; see Figure 8). Population Model 2 (PM2) was a mediation model adopted from an empirical study (Wei et al., 2003; see Figure 9). Population Model 3 (PM3) was a multiple-indicator cross-lagged panel model (Hamaker et al., 2015; Rogosa, 1980; see Figure 10). To take into account the wording effect, item residuals from the same item across three time points were allowed to covary (Little, 2013).

Figure 8 *Path Diagram of Population Model 1*

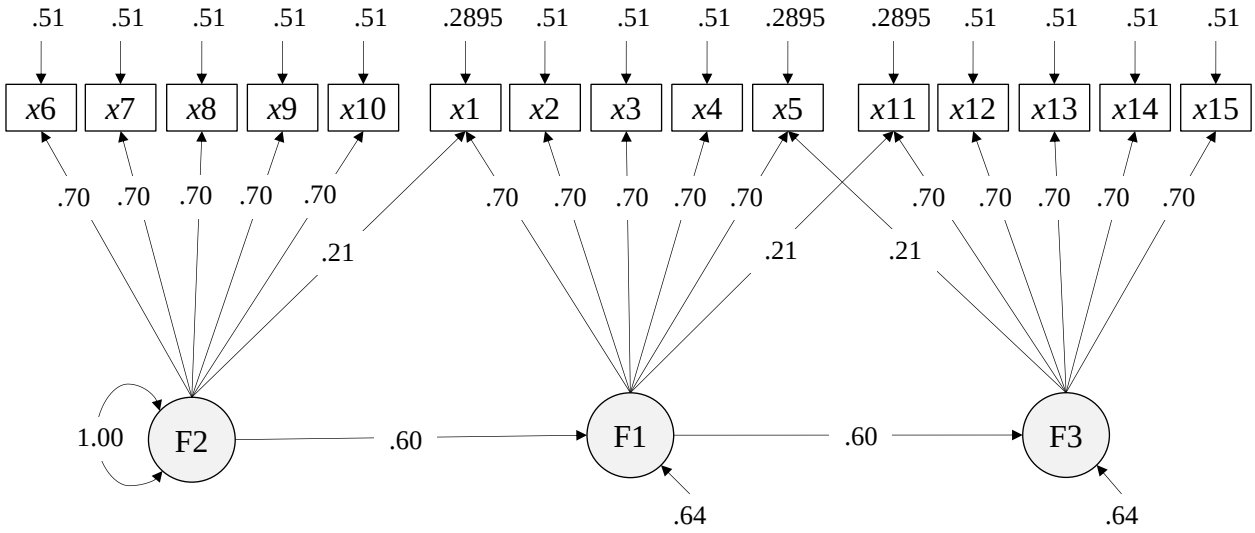


Figure 9 *Path Diagram of Population Model 2*

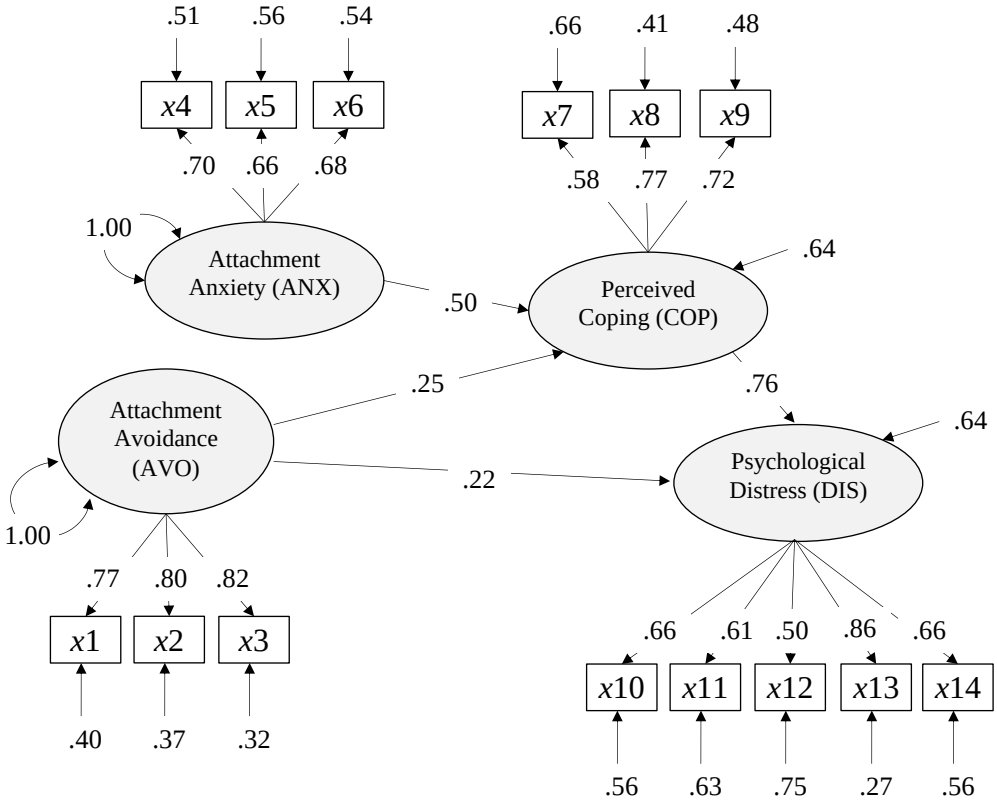
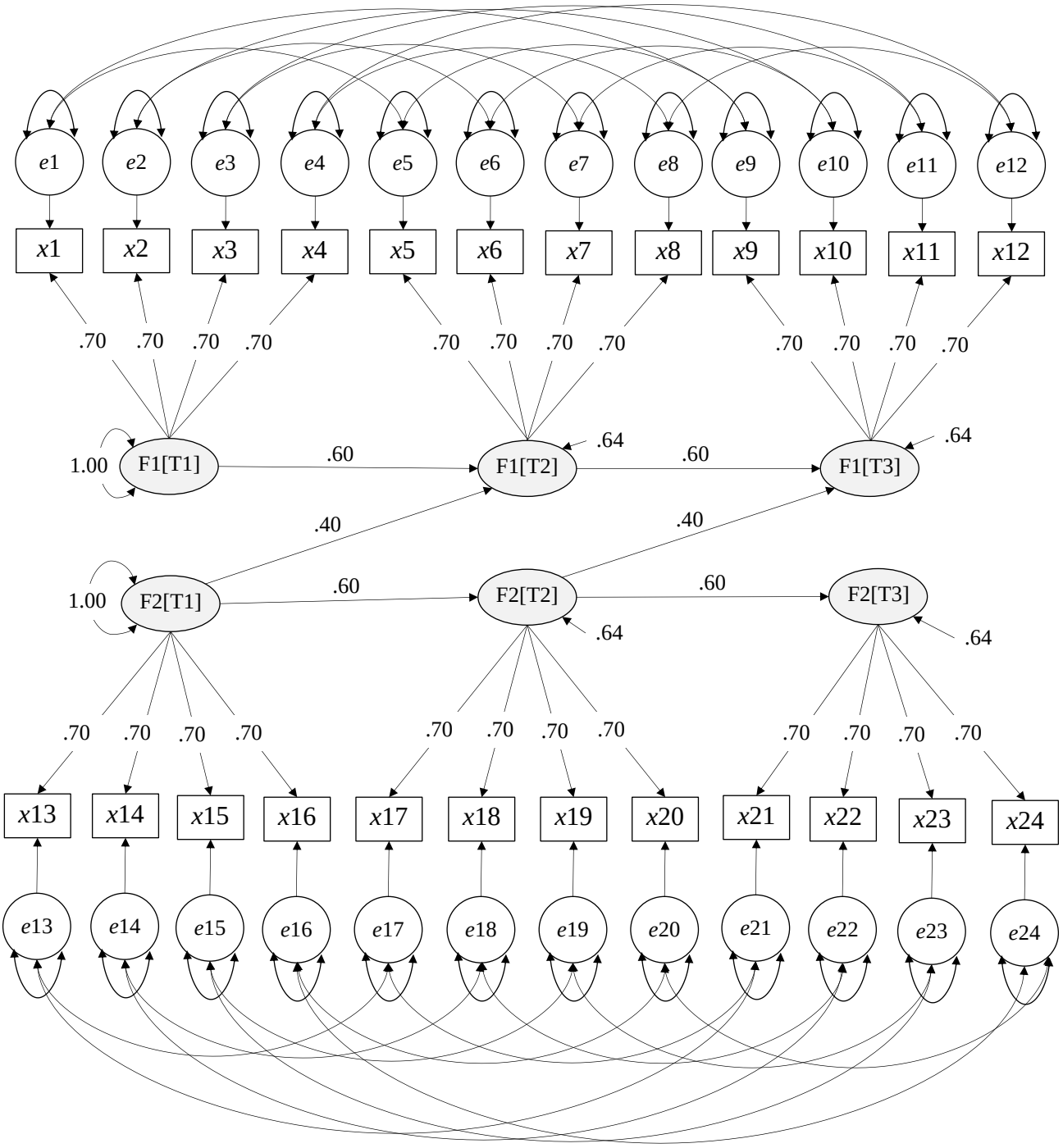


Figure 10 *Path Diagram of Population Model 3*



Note: "T1" / "T2" / "T3" in square brackets denote time point 1 / 2 / 3. The population value of item residual variances and covariances is set to 0.51 and 0.10, respectively.

4.1.2 Data Missingness

The simulation considered two typical patterns of data missingness: i) "complete data" where no data is missing, and ii) incomplete data with 10% missing completely at random (MCAR). Missing at random (MAR) and not missing at random (NMAR) were excluded to keep the scale of the simulation study manageable.

4.1.3 Data Non-normality

We planned to test four severity level of data non-normality which were i) normally distribution ("normality"), ii) mild non-normality, iii) moderate non-normality, and iv) severe non-normality. Based on a systematic review on 194 empirical studies summarized by Cain and colleagues (2017), we drew on the 25^{th} / 50^{th} / 75^{th} percentile of multivariate skewness and multivariate kurtosis (see Cain et al., 2017, Table 2) to represent the mild / moderate / severe non-normality in this simulation study. The value of multivariate skewness and multivariate kurtosis specified for the population models is shown in Table 7.

Table 7 *Multivariate Skewness and Multivariate Kurtosis*

Population Model	Non-normality	Multivariate skewness of observed variables	Multivariate skewness of latent variables	Multivariate kurtosis of observed variables	Multivariate kurtosis of latent variables
PM1	Normal	0	0	255	15
	Mild	1	1	287	47
	Moderate	3	3	316	76
	Severe	15	15	346	106
PM2	Normal	0	0	224	24
	Mild	1	1	256	56
	Moderate	3	3	285	85
	Severe	15	15	315	115
PM3	Normal	0	0	624	48
	Mild	1	1	656	80
	Moderate	3	3	685	109
	Severe	15	15	715	139

Note: The values of multivariate kurtosis shown are uncentered, which can be centered on p (p + 2) where p is the number of variables.

4.1.4 Sample size

We considered four sample sizes: 100, 200, 500 and 1,000. Sample size of 100 is typically required as the minimum sample size for conducting SEM analysis. Sample size up to 1,000 symbolizes an infinitely large sample.

The whole simulation study consisted of a total of 32 conditions: (i) two data missingness (complete data & incomplete data with 10% MCAR), (ii) four data non-normalities (normal, mild, moderate & severe) and (iii) four sample sizes (100, 200, 500 & 1,000). Under each of the 32 simulation conditions, we generated 500 artificial datasets based on the configuration of three population models.

4.2 Data Generation

Data generation was implemented in R (v4.1.0; R Core Team, 2021). We first generated the scores on latent variables, followed by the scores on observed variables. The variances in latent exogenous variables and the error terms of latent endogenous variables were simulated with a R function of "mnonr" from the R package of "mnonr" (v1.0.0; Qu & Zhang, 2020) which allows for the specification of the expected values of multivariate skewness and of multivariate kurtosis together with the sample size (Qu et al., 2020). Then the scores on latent endogenous variables were computed with equation 1.2. After that, item residual variances, or the variances in the residuals of observed item indicators, were simulated with the same function of "mnonr" where we input the corresponding expected values of multivariate skewness and kurtosis, from which the scores on observed variables would result using equation 1.1. Subsequently, we collated the scores on observed variables into a matrix form while abandoning the scores on latent variables. At last, to simulate the two patterns of data missingness, we took the multivariate amputation approach (Schouten et al., 2018). For the generated datasets with 10% MCAR, we ran the "ampute" function in the R package of "mice" (v3.13.0; van Buuren, 2021) wherein we specified the missingness

proportion as 0.1 and the missing data patterns as “MCAR” in. For the generated datasets with no missing data, we skipped the “ampute” function. As a quality check, the generated datasets were fitted into the correctly specified measurement models with CFA in lavaan (v0.6.8; Rosseel, 2012). Of those running into improper and nonconverging solutions, we discarded them. Following the common practice of simulation studies (e.g., MacKinnon et al., 1995), we generated 500 datasets per condition. The generated dataset was then analysed by each of the LINGAM-LV algorithms. Please refer to Appendix 16 for the R codes for data generation.

4.3 Data Analysis

In PM1, 500 artificial datasets were generated for each of the 32 simulation conditions. Each of the 500 artificial datasets went through the following. With CFA, the measurement model of PM1 was first fitted into the artificial dataset, from which we obtained a factor loading matrix and a factor covariance matrix. Then, the artificial dataset together with the two matrices were then entered into DirectLiNGAM-LV and ParceLiNGAM-LV. The artificial dataset and the factor loading matrix were also fed into ICA-LiNGAM for it does not require a factor covariance matrix in its computation. With the input in place, we executed the three LiNGAM-LV algorithms, each of which output a discovered path model. The above procedure was repeated for each of the 500 replications. We gathered all the discovered path models by the three algorithms. It allowed us to evaluate the performance of the algorithms by comparing the configuration of the discovered path models against the actual model (or the path model of PM1). For those discovered models that completely matched the actual model, we went on to compare the parameter estimates of the pathways in the discovered models with the population value of the pathways in the actual model. We repeated the same operation on PM2 and PM3. LiNGAM algorithms were prepared and are implementable in R, the code for which can be found in Appendix 16. For R documentation

on LiNGAM algorithms, please refer to Appendix 12. Table 8 summarizes hyperparameters the algorithms used in the simulation.

Table 8 *Hyperparameters Specified in the Simulation Study*

Hyperparameters	Specified value
Penalty type	"alasso"
Number of penalty value to test	10
Change in penalty values per iteration	.03
For ICA-LiNGAM-LV only	
Type of ICA	"fastICA"
Maximum iteration of ICA	1000

4.4 Performance Criteria

Making comparison of the discovered models by the algorithms against the actual models allows us to evaluate the performance of the algorithms. To see how close the configuration of the discovered model is to that of the actual model, we used path-related fit indices and accuracy as performance indicators. For those discovered path models that exactly match the actual path model, we used root mean square error to examine the difference between the parameter estimates of the pathways by the algorithms and the population values from the actual model.

4.4.1 Path-related Fit Indices

RMSEA-P and CFI-P were to monitor the fitness of the discovered path models. According to the recommended cut-off values by Williams and O'Boyle (2011), RMSEA-P below or equal to 0.05 indicates a close fit, between 0.05 and 0.08 as a reasonable fit, between 0.08 and 0.10 as a mediocre fit, and above 0.10 as a poor fit. For CFI-P, we followed Lance et al. (2016)'s guideline that CFI-P greater than 0.99 is interpreted as a perfect fit.

4.4.2 Accuracy

Between the discovered path model by the algorithms and the actual path model, a (non-zero) pathway can be classified into one of the four possibilities: true positive, true negative, false positive and false negative. True positive (TP) happens when the pathway is

present in both models. True negative (TN) happens when the pathway is absent from both models. False positive (FP) happens when the pathway is present in the discovered model but not in the actual model. False negative (FN) happens when the pathway is absent from the discovered model but is present in the actual model. Table 9 summaries the four possibilities into a (2 × 2) confusion matrix. TP, TN, FP and FN are elemental to the computation of evaluation metrics for binary classification tasks. Accuracy is a typical evaluation metric that this thesis reports.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{4.1}$$

Table 9 *Confusion Matrix for Binary Classifier on a Pathway*

		Discovered model	
		Presence of pathway	Absence of pathway
Actual model	Presence of pathway	True positive	False negative
	Absence of pathway	False positive	True negative

4.4.3 Root Mean Square Error

RMSE is the discrepancy in the estimates of structural parameters between the discovered and the actual model, provided that the accuracy of the discovered model reaches 100%. RMSE is expressed in Forbenius norm as:

$$\text{RMSE} = \sqrt{\text{tr}\left\{\left(\mathbf{B}_{\text{Actual}} - \widehat{\mathbf{B}}\right)^{\text{T}}\left(\mathbf{B}_{\text{Actual}} - \widehat{\mathbf{B}}\right)\right\}} \tag{4.2}$$

where “tr” refers to the trace of a square matrix, $\widehat{\mathbf{B}}$ is the structural parameter matrix of the discovered model, and $\mathbf{B}_{\text{Actual}}$ is the structural parameter matrix of the actual model.

Chapter 5: Results

5.1 Preliminary Analysis

A correctly specified measurement model is a prerequisite of running LiNGAM-LV algorithms. To check if the measurement models specified in this simulation study were correct, we make use of two normative fit indices: global RMSEA and global CFI. Figures 11 and 12 plot the mean values of the global RMSEA and of the global CFI by condition and by population model over 500 replications, from which the values of RMSEA were below the standard cut-off values of 0.05 (Hu & Bentler, 1999) and the values of CFI were above the cut-off of 0.95 (Hu & Bentler, 1999). It suggests acceptable fits of the measurement models and thus a successful data generation.

Figure 11 *Global RMSEA of Measurement Models over 500 Replications*

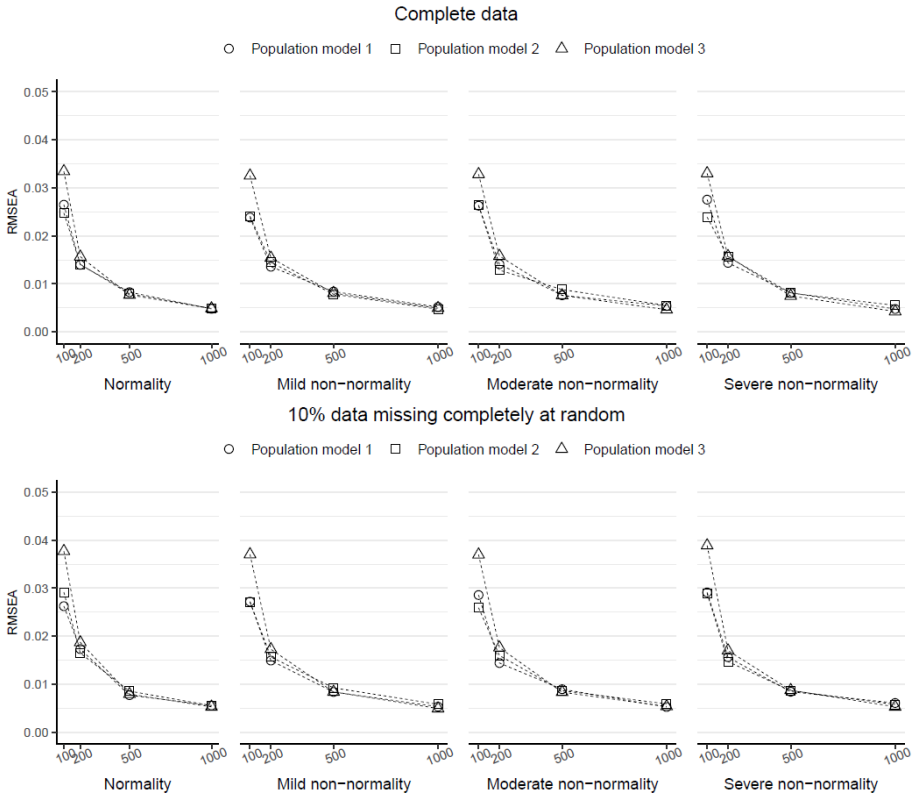
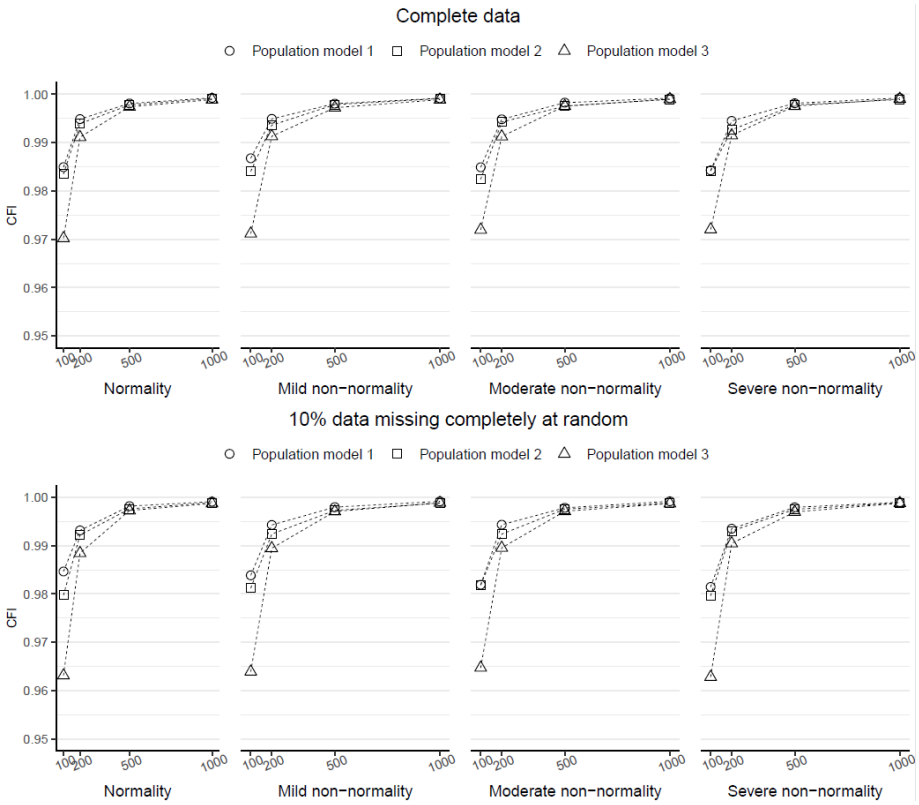


Figure 12 *Global CFI of Measurement Models over 500 Replications*



5.2 Population Model 1

PM1 was a causal chain model with three latent variables. In general, the discovered models by the algorithms yielded acceptable fits. Figure 13 illustrates the mean values of RMSEA-P of the discovered path models over 500 replications per condition, from which the range of RMSEA-P was from 0.009 to 0.025, suggesting a close fit. We observe a slight decline and improvement in RMSEA-P with the increase in sample size. When the sample size was at its smallest of 100, the values of RMSEA-P reached its highest. When the sample size was as large as 1,000, the values of RMSEA-P dropped to its lowest. Data missingness and data non-normality conditions virtually had no impact on the change in RMSEA-P. RMSEA-P of the discovered path models by the three algorithms were close to one another, with their discrepancies further narrowing down with the increase in sample size. The values of RMSEA-P of the algorithms overlapped when the sample size was 1,000.

Figure 13 *RMSEA-P on Population Model 1*

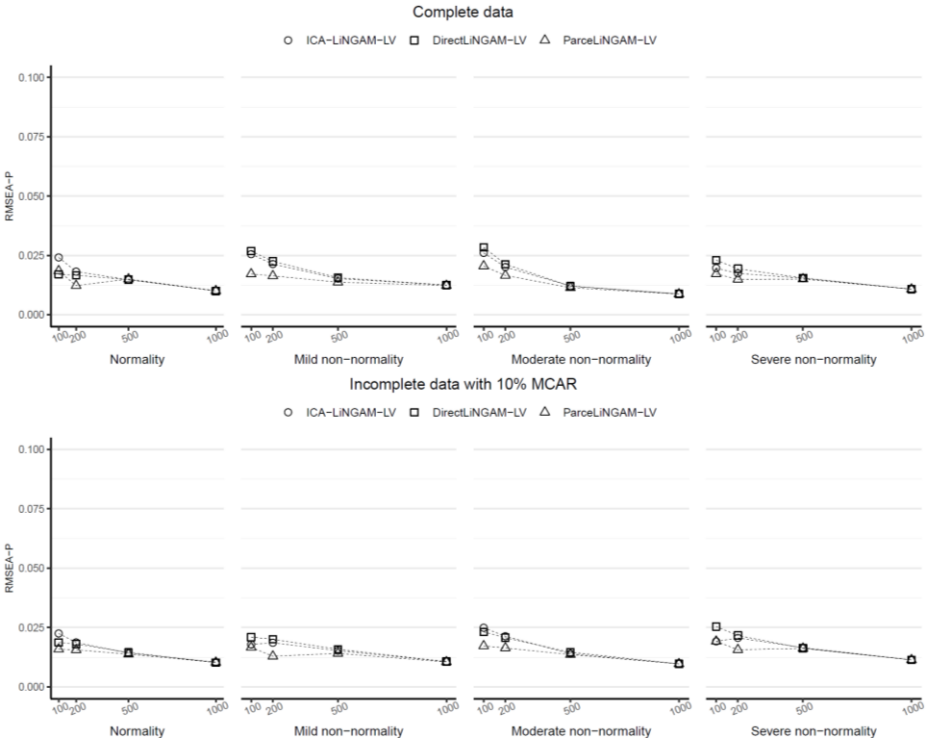


Figure 14 illustrates the mean values of CFI-P which ranged from 0.992 to 0.999 across the algorithms, suggesting a perfect fit. With the increase in sample size, the values of CFI-P mildly went up and improved. The value of CFI-P was virtually the same for the three algorithms. Figure 15 plots the mean value of accuracy over 500 replications which ranged from 0.704 to 1. In general, the accuracy level increased and improved with the rise in sample size. The accuracy level attained its lowest when the sample size was as small as 100. The accuracy level reached its highest when the sample size was as high as 1,000. Data missingness and data non-normality conditions had little impact on the change in accuracy level with the increase in sample size. We observed one exception, however. When the sample size was at 1,000 and data non-normality was severe, the accuracy of the discovered path model by ParceLiNGAM-LV uncharacteristically declined and did not follow the same raising trend as the other two algorithms did. We will revisit such pattern of ParceLiNGAM-LV closely in the discussion section. Amongst the three algorithms, the accuracy of the

discovered path models by ICA-LiNGAM-LV came close to that of DirectLiNGAM-LV. Yet, the accuracy of ParcelLiNGAM-LV was the lowest among the three.

Figure 14 *CFI-P on Population Model 1*

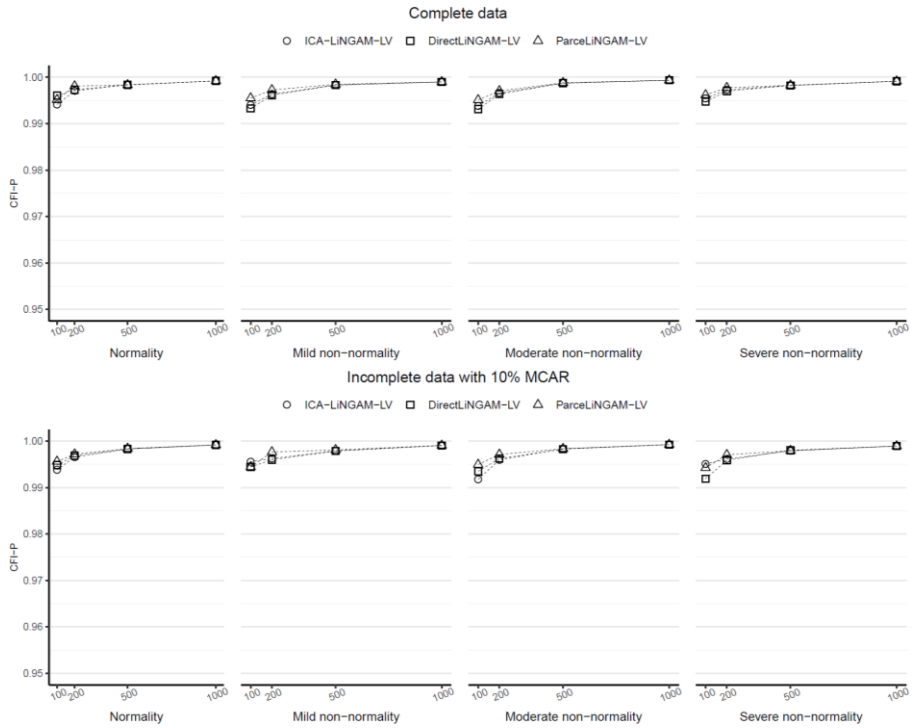


Figure 15 *Accuracy on Population Model 1*

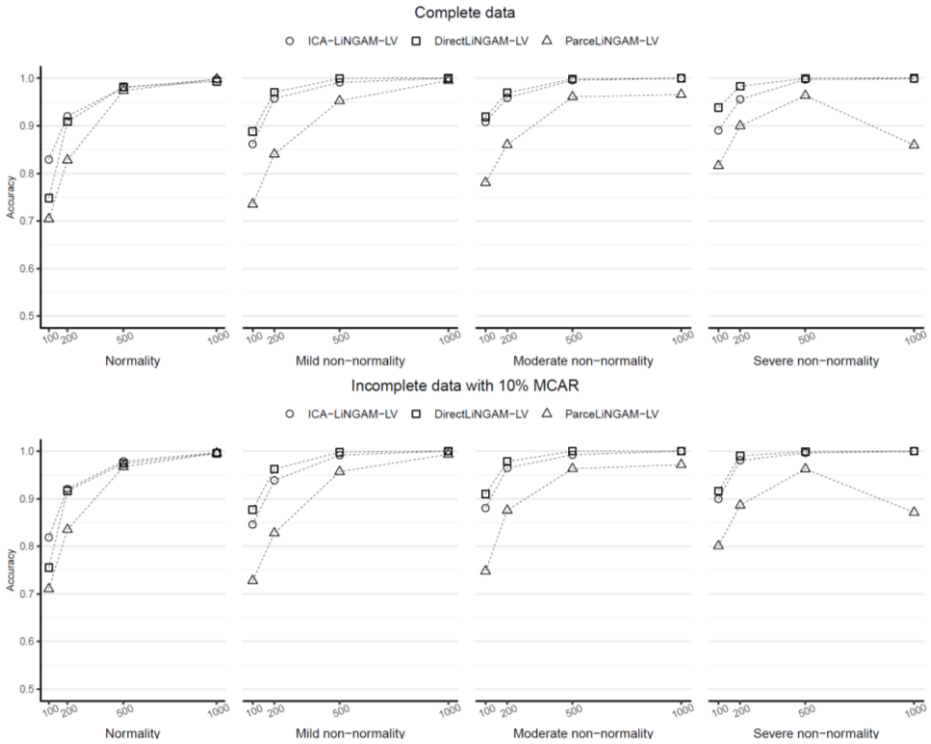
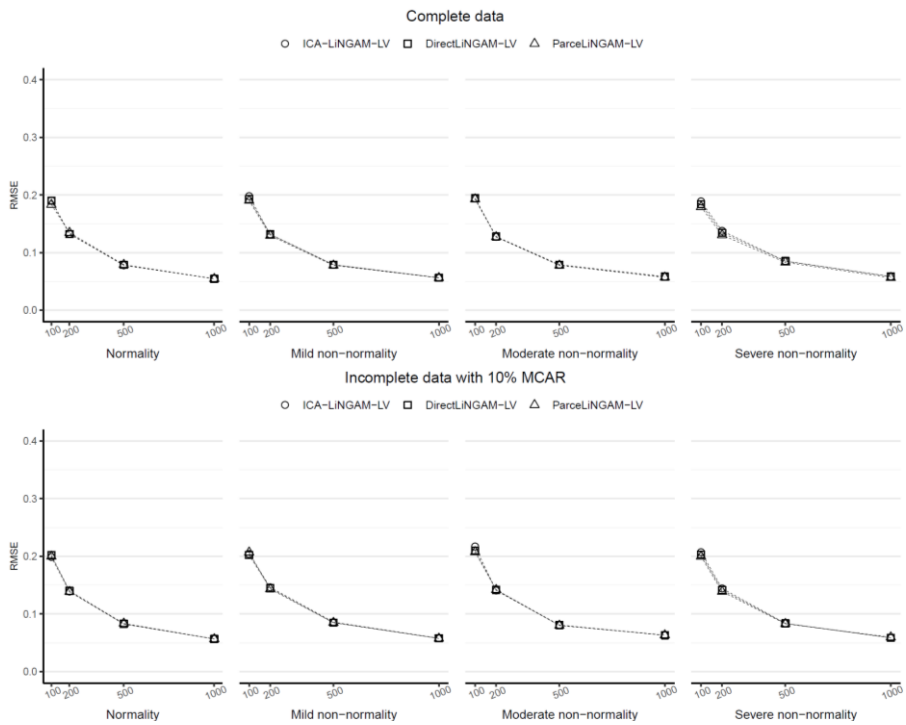


Figure 16 depicts the mean of RMSE on PM1 over the replications that produced the discovered path model that exactly matched the actual model. The range of RMSE was from 0.031 to 0.217. The highest (and poorest) RMSE happened when the sample size was 100 and the data was normal with no missing value. Overall, RMSE dropped and improved when sample size increased from 100 to 1,000. We observed no difference in RMSE between the algorithms.

Figure 16 *Root Mean Square Error on Population Model 1*

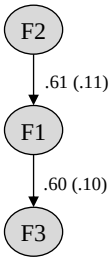


On a practical level, we are interested in the likelihood that a discovered model completely matches the actual path model; if they do not match, we reckon what types of mistakes the algorithms would commit. To give an overview, we gathered a pool of 16,000 (= 500 replications × 32 conditions) discovered path models. From the pool we listed three of the most frequently models being discovered in Figure 17. In it, the numeric values outside (within) the parenthesis next to the arrow refer to the mean (standard deviation) of the parameter estimates of the respective pathway. The absence of pathways assumes zero values of the structural parameters. DirectLiNGAM-LV had 89.5% chance of discovering the true

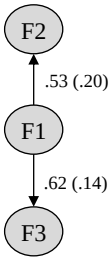
model, seconded by ICA-LiNGAM-LV (87.9%) and ParceLiNGAM-LV (72.2%). From Figure 17, there were less than 30% of the discovered path models from three algorithm that did not match the actual path models in full. Of them, we spotted three typical errors. A wrong directed edge extended from F1 to F2 which should have been from F2 to F1; a wrong directed edge extended from F3 to F1 which should have been from F1 to F3; a redundant pathway was added from F2 to F1. Yet, the occurrence of these errors was relatively low on PM1. Moreover, the likelihood of discovering the actual model increases with the increase in sample size. Appendix 14 shows for the top three most frequently discovered path models by the algorithms on population models by sample size.

Figure 17 *Three Most Frequently Discovered Path Models on Population Model 1*

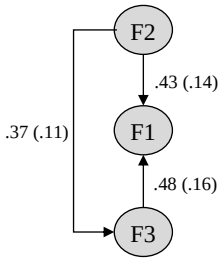
ICA-LiNGAM-LV



1st: 87.89% (*n* = 14061)
RMSEA-P: *M* = .02; *SD* = .04
CFI-P: *M* = 1.00; *SD* = .01
(Complete match)

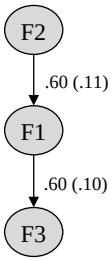


2nd: 6.83% (*n* = 1093)
RMSEA-P: *M* = .00; *SD* = .00
CFI-P: *M* = 1.00; *SD* = .00
(F2-F1 pathway in wrong direction)

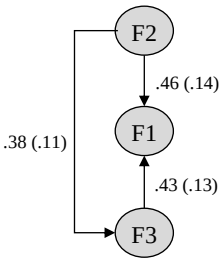


3rd: 2.19% (*n* = 351)
RMSEA-P: *M* = .00; *SD* = .00
CFI-P: *M* = 1.00; *SD* = .00
(F2-F3 pathway extra & F1-F3 pathway in wrong direction)

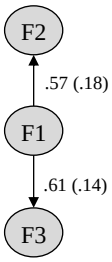
DirectLiNGAM-LV



1st: 89.52% (*n* = 14323)
RMSEA-P: *M* = .02; *SD* = .04
CFI-P: *M* = 1.00; *SD* = .01
(Complete match)

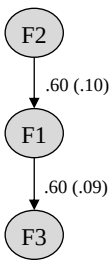


2nd: 3.73% (*n* = 598)
RMSEA-P: *M* = .00; *SD* = .00
CFI-P: *M* = 1.00; *SD* = .00
(F2-F3 pathway extra & F1-F3 pathway in wrong direction)

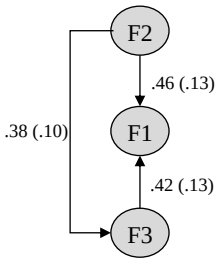


3rd: 2.72% (*n* = 435)
RMSEA-P: *M* = .00; *SD* = .00
CFI-P: *M* = 1.00; *SD* = .00
(F2-F1 pathway in wrong direction)

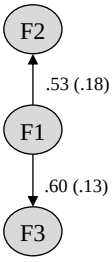
ParceLiNGAM-LV



1st: 72.20% (*n* = 11552)
RMSEA-P: *M* = .02; *SD* = .04
CFI-P: *M* = 1.00; *SD* = .01
(Complete match)



2nd: 9.99% (*n* = 1599)
RMSEA-P: *M* = .00; *SD* = .00
CFI-P: *M* = 1.00; *SD* = .00
(F2-F3 pathway extra & F1-F3 pathway in wrong direction)



3rd: 7.01% (*n* = 1121)
RMSEA-P: *M* = .00; *SD* = .00
CFI-P: *M* = 1.00; *SD* = .00
(F2-F1 pathway in wrong direction)

Note: The percentage refers to the likelihood that the corresponding path model being discovered, equivalent to n / 16,000 × 100%, where n = is the number of simulated datasets that were fed to the algorithm to give the corresponding path model; 16,000 is the total number of simulated datasets (= 2 data missingness × 4 data non-normality × 4 sample sizes × 500 replications). Above each of the arrows, the number outside the parenthesis refers to the average of the parameter estimates (over n) and the number within the parenthesis refers to the standard deviation of the parameter estimates. M = mean; SD = standard deviation.

5.3 Population Model 2

PM2 was a mediation model with four latent variables: attachment anxiety (ANX), attachment avoidance (AVO), perceived coping (COP) and psychological distress (DIS). It has one more latent variable than PM1. Figure 18 shows the mean of RMSEA-P for three algorithms. The range of RMSEA-P was 0.009 to 0.071. When the sample size was as small as 100, the value of RMSEA-P was its highest above the standard threshold of 0.05, a sign of a reasonable fit. With the increase in sample size, RMSEA-P dropped below 0.05 level, indicating a close fit. Also, when the sample size was small, the level of data non-normality had some slight detrimental effect on RMSEA-P. Provided the sample size of 100, the more severe the non-normality was, the higher the value of RMSEA-P was. We found no conspicuous effect of data missingness on the values of RMSEA-P. Types of the algorithms played a role in RMSEA-P. While controlling for data missingness, non-normality and sample size, the values of RMSEA-P by ParceLiNGAM-LV were systematically higher and worse than those of ICA-LiNGAM-LV and DirectLiNGAM-LV. The values of RMSEA-P by ICA-LiNGAM-LV and DirectLiNGAM-LV virtually overlapped.

Figure 18 *RMSEA-P on Population Model 2*

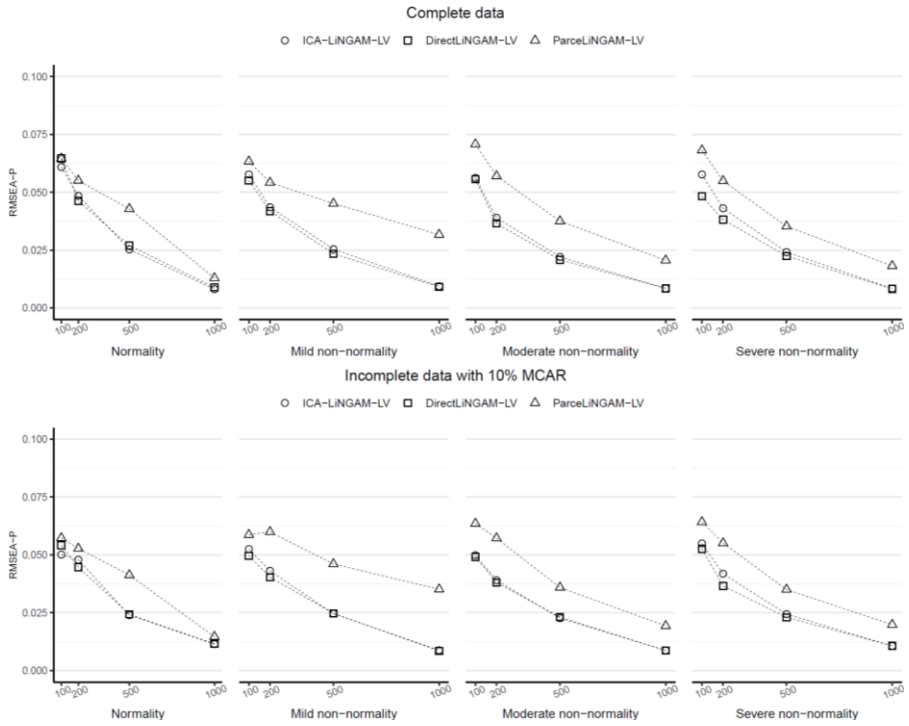


Figure 19 shows the mean of CFI-P on PM2. The range of CFI-P was from 0.966 to 0.999. In the smallest-possible sample size of 100, the values of CFI-P hit their lowest point, suggesting a poor fit. When sample size went up, the values of CFI-P gradually increased. When sample size was 500 or above, the values of CFI-P passed through the cut-off of 0.99, suggesting a perfect fit. Conditions of data missingness and of non-normality did not have a visible impact on the change in CFI-P. However, the choice of algorithms did have. The values of CFI-P by ParceLiNGAM-LV was consistently lower and worse than those of ICA-LiNGAM-LV and DirectLiNGAM-LV, while we controlled for all other simulation conditions. The values of CFI-P by ICA-LiNGAM-LV were virtually identical to those of DirectLiNGAM-LV.

Figure 19 *CFI-P on Population Model 2*

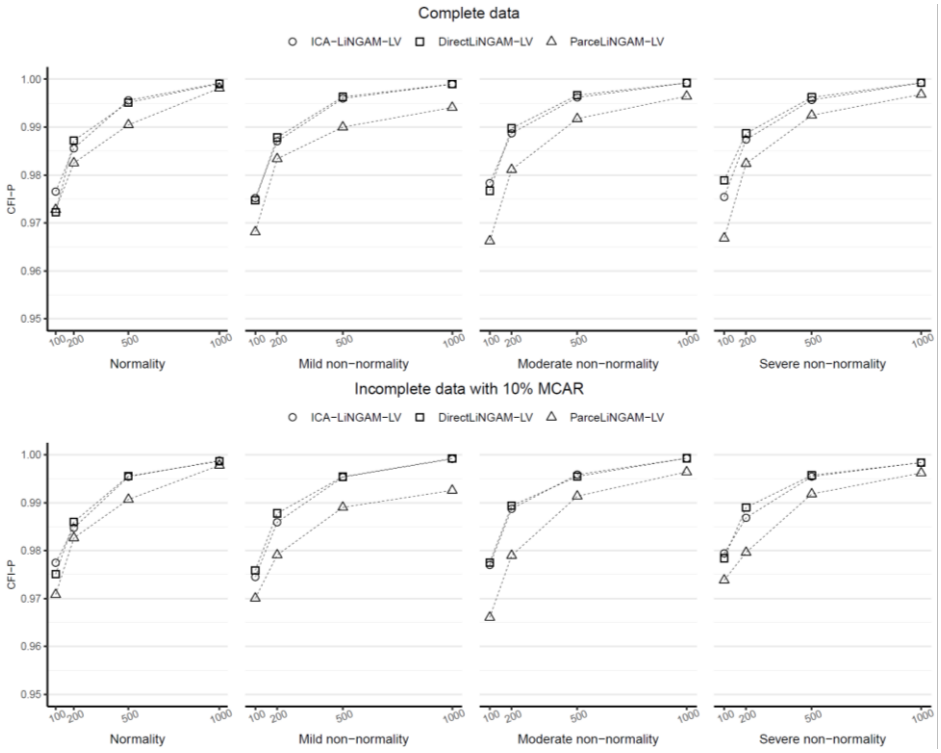


Figure 20 shows the accuracy mean of three algorithms on PM2 where the lowest value of accuracy was 0.610 and the highest was 0.963. Not a single condition brought the accuracy level to 100%. Sample size had a substantial influence on accuracy. When the sample size was at its smallest, the accuracy level dropped to its lowest value. With the increase in sample size, the accuracy level climbed steadily until the sample size was 500. When sample size passed through 500 to 1,000, the accuracy level plateaued out and was capped at around 0.9. An increase in the non-normality level mildly pushed up the rate of change in accuracy with sample size. We found data missingness of no apparent impact on accuracy. The accuracy of the discovered path models by ParceLiNGAM-LV was the poorest amongst the three. The accuracy of ICA-LiNGAM-LV and of DirectLiNGAM-LV was virtually indistinguishable.

Figure 20 *Accuracy on Population Model 2*

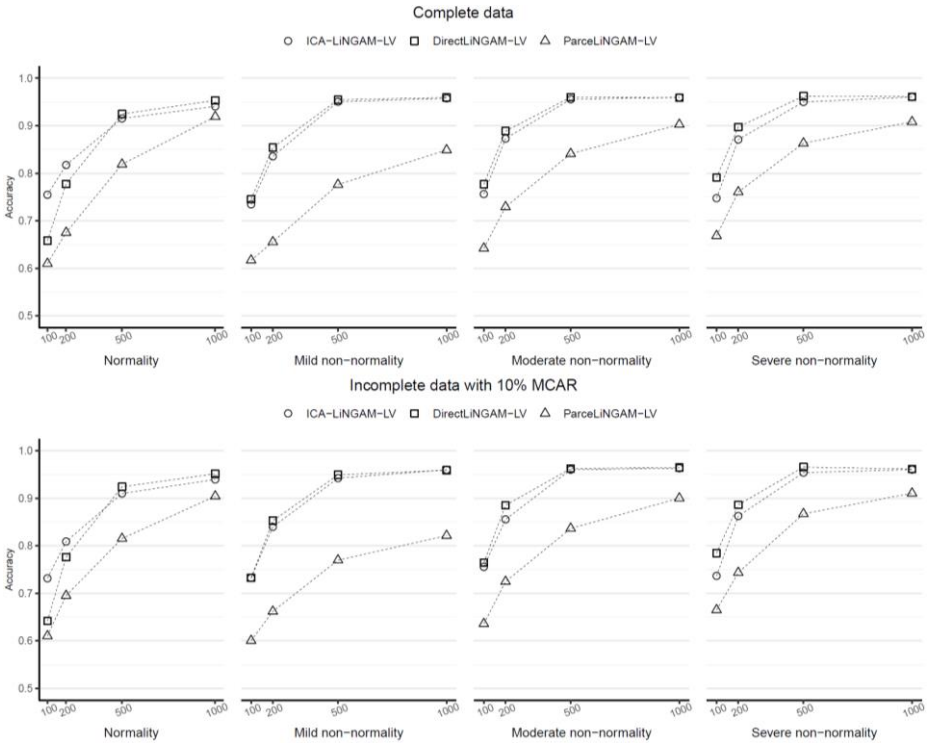
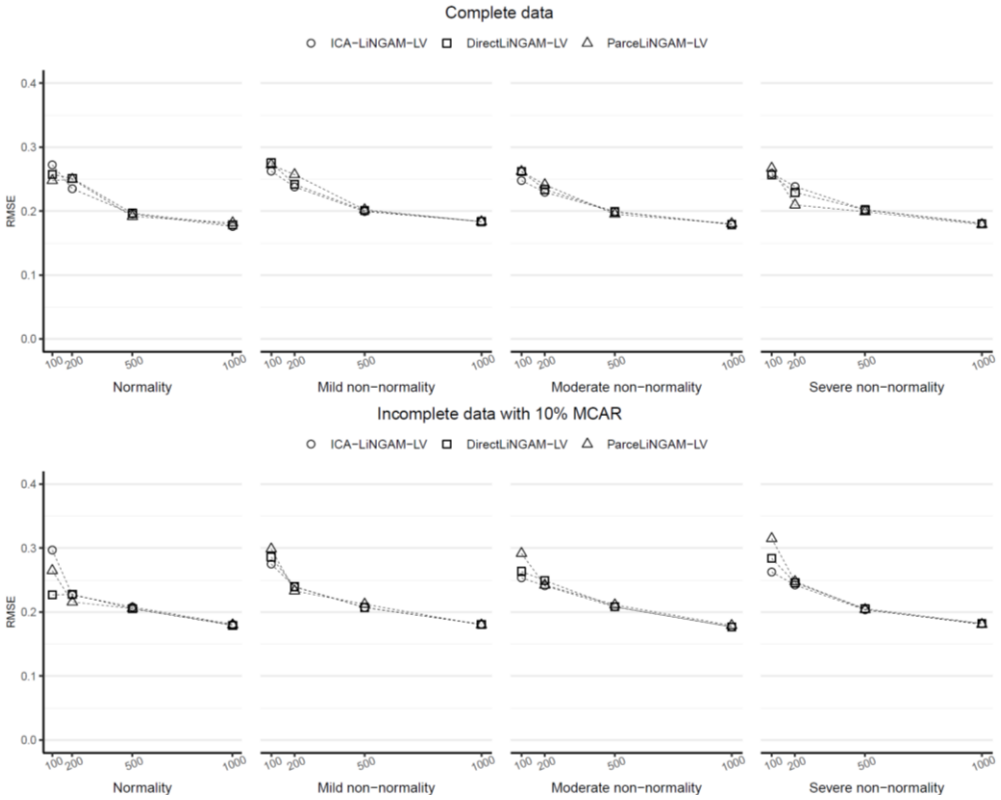


Figure 21 shows the mean of RMSE on PM2. The range of RMSE was 0.176 to 0.299. An increase in sample size led to the decline in RMSE. RMSE was at its greatest when the sample size was as small as 100. RMSE was its lowest at the sample size of 1,000. The severity of non-normality mildly sped up the decline in RMSE with sample size. Yet we witness no effect of data missingness on RMSE. RMSEs were very close between the algorithms.

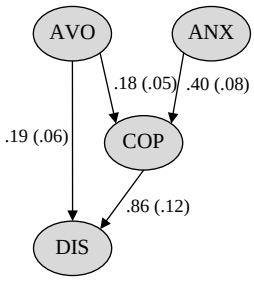
Figure 21 *RMSE on Population Model 2*



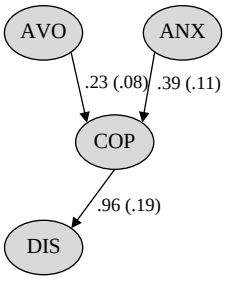
Regarding the likelihood of discovering the actual model, the summary is in Figure 22. Models discovered by DirectLiNGAM-LV had 37.2% success rate of hitting the actual path model, followed by ICA-LiNGAM-LV (35.9%) and ParceLiNGAM-LV (19.3%). Some common mistakes made by the algorithms included the missing pathway from AVO to DIS and a redundant pathway from AVO to ANX.

Figure 22 *Three Most Frequently Discovered Path Models on Population Model 2*

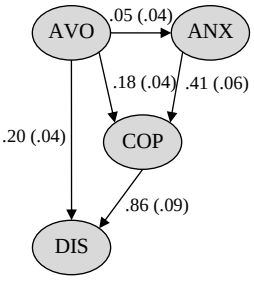
ICA-LiNGAM-LV



1st: 35.93% (*n* = 5749)
RMSEA-P: *M* = .01; *SD* = .03
CFI-P: *M* = 1.00; *SD* = .00
(Complete match)

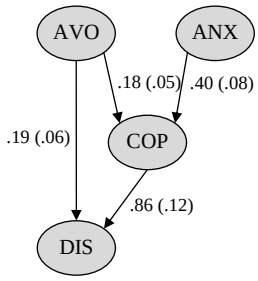


2nd: 13.73% (*n* = 2198)
RMSEA-P: *M* = .06; *SD* = .04
CFI-P: *M* = .98; *SD* = .03
(AVO-DIS pathway missing)

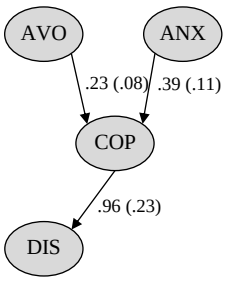


3rd: 12.81% (*n* = 2050)
RMSEA-P: *M* = .01; *SD* = .02
CFI-P: *M* = 1.00; *SD* = .00
(AVO-ANX pathway extra)

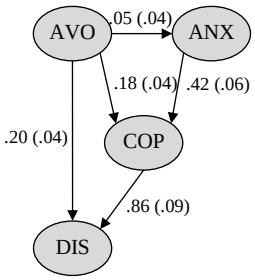
DirectLiNGAM-LV



1st: 37.21% (*n* = 5954)
RMSEA-P: *M* = .01; *SD* = .03
CFI-P: *M* = 1.00; *SD* = .00
(Complete match)

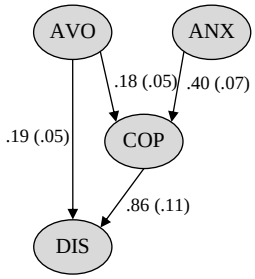


2nd: 13.74% (*n* = 2199)
RMSEA-P: *M* = .06; *SD* = .04
CFI-P: *M* = .98; *SD* = .03
(AVO-DIS pathway missing)

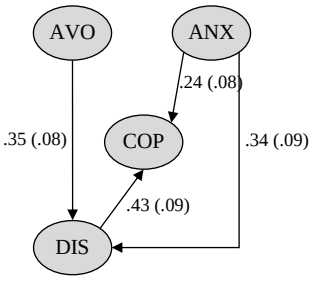


3rd: 7.34% (*n* = 1175)
RMSEA-P: *M* = .01; *SD* = .02
CFI-P: *M* = 1.00; *SD* = .00
(AVO-ANX pathway extra)

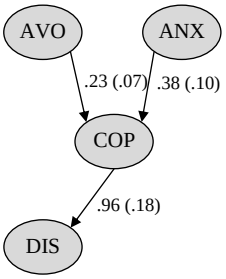
ParceLiNGAM-LV



1st: 19.33% (*n* = 3092)
RMSEA-P: *M* = .01; *SD* = .03
CFI-P: *M* = 1.00; *SD* = .00
(Complete match)



2nd: 6.40% (*n* = 1024)
RMSEA-P: *M* = .02; *SD* = .03
CFI-P: *M* = 1.00; *SD* = .01
(ANX-DIS pathway extra, AVO-COP pathway missing, and COP-DIS pathway in wrong direction)



3rd: 5.68% (*n* = 908)
RMSEA-P: *M* = .06; *SD* = .04
CFI-P: *M* = .98; *SD* = .01
(AVO-DIS pathway missing)

5.4 Population Model 3

PM3 was a cross-lagged panel model with latent variables. Figure 23 depicts the mean value of RMSEA-P on PM3. RMSEA-P ranged from 0.009 to 0.057. With the increase in sample size, RMSEA-P gradually dropped and improved. When the sample size was as low as 100, RMSEA-P was above the 0.05 threshold, suggesting a reasonable fit. When the sample size went up to 1,000, RMSEA-P declined and passed through the 0.05 threshold, suggesting a close fit. The non-normality condition had an impact of the rate of change in RMSEA-P with sample size. Specifically, the more severe the non-normality of the data was, the higher the rate of decrease in RMSEA-P with the increase of sample size. Between the algorithms, we observed RMSEA-P of DirectLiNGAM-LV was the lowest (and the greatest). ICA-LiNGAM-LV was the second lowest, and ParceLiNGAM-LV was the highest.

Figure 23 RMSEA-P on Population Model 3

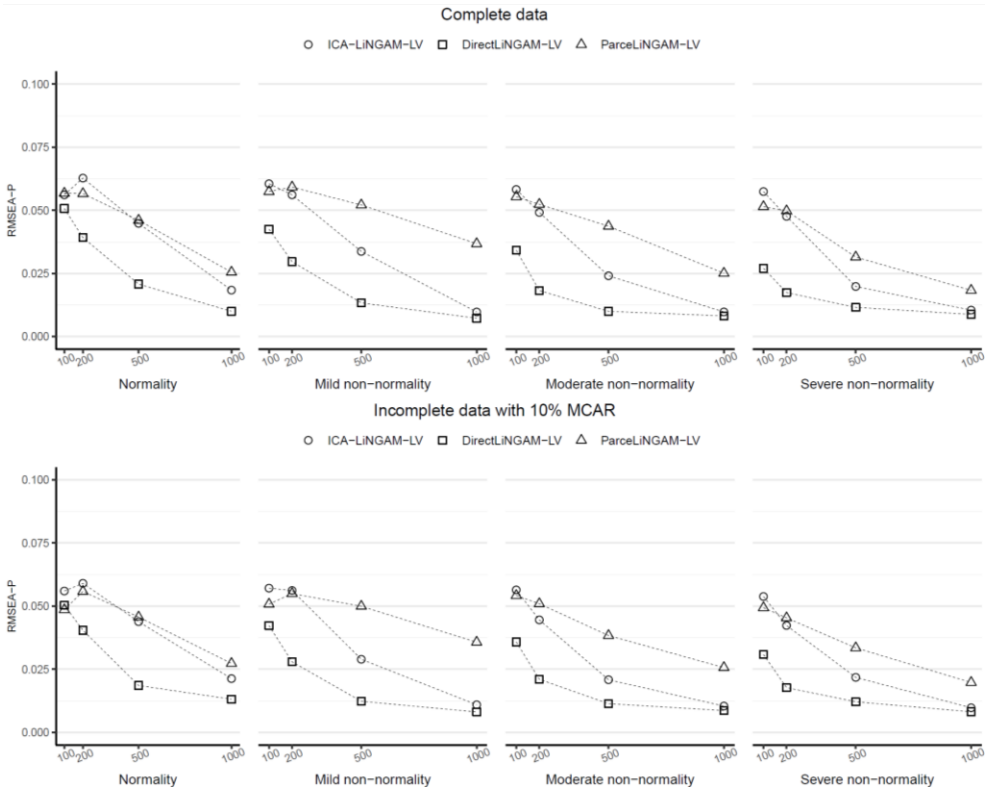


Figure 24 shows the mean value of CFI-P on PM3. The range of CFI-P was from 0.962 to 0.999. Overall, the rise in sample size brought up the values of CFI-P. When the

sample size was at its smallest of 100, CFI-P was underneath the .99 threshold, suggesting a poor fit. When the sample size was at its greatest of 1,000, CFI-P passed through the .99 threshold, an indication of an almost perfect fit. The severity of data non-normality influenced the rate of change in CFI-P with sample size. When the non-normality of the data became increasingly severe, CFI-P climbed up at a higher rate with the increase in sample size. Between the algorithms, CFI-P of DirectLiNGAM-LV was apparently higher than those of ICA-LiNGAM-LV and of ParceLiNGAM-LV. Whether CFI-P of ICA-LiNGAM-LV was better than that of ParceLiNGAM-LV depended on the sample size. When the sample size was under 200, ParceLiNGAM-LV outperformed ICA-LiNGAM-LV in CFI-P. When the sample size rose above 200, the lead reversed and this time ICA-LiNGAM-LV outperformed ParceLiNGAM-LV.

Figure 24 *CFI-P on Population Model 3*

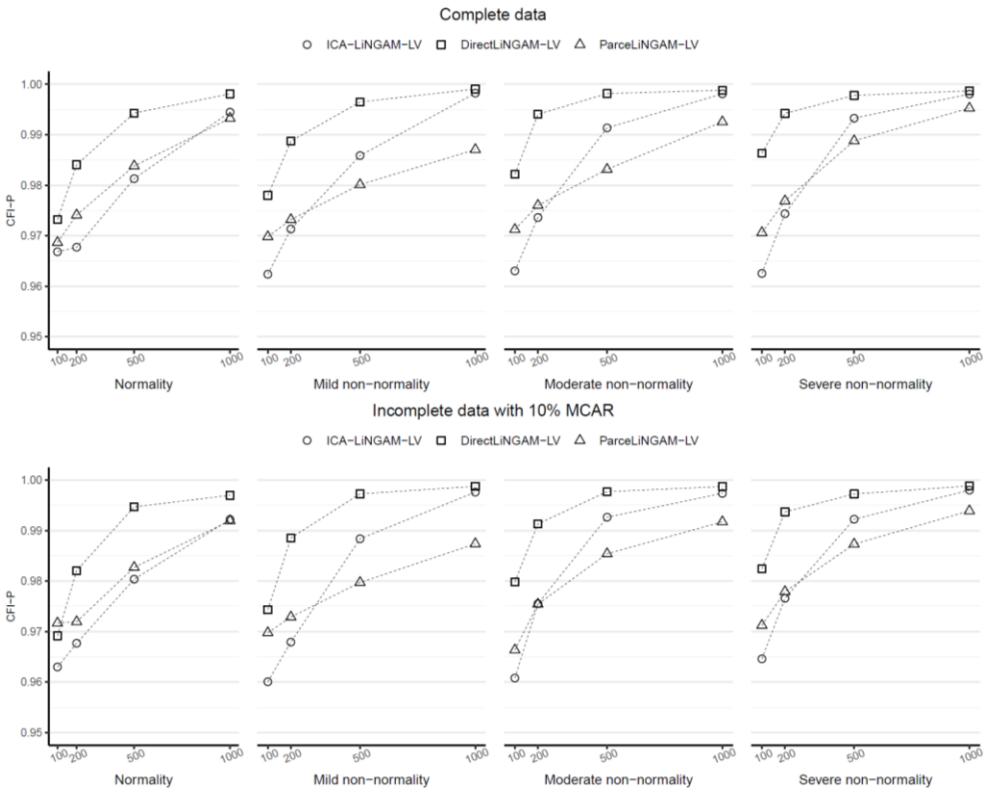


Figure 25 shows the mean of accuracy on PM3. The range of accuracy was from 0.665 to 0.997. As usual, the accuracy level rose and improved with the increase in sample

size. The data non-normality affected the rate of improvement in accuracy. In particular, the more severe the non-normality, the higher the rate at which the accuracy improved with the increase of sample size. Between the algorithms, the accuracy of the discovered path models by DirectLiNGAM-LV was systematically the highest, followed by ICA-LiNGAM-LV. ParceLiNGAM was in the last place.

Figure 25 *Accuracy on Population Model 3*

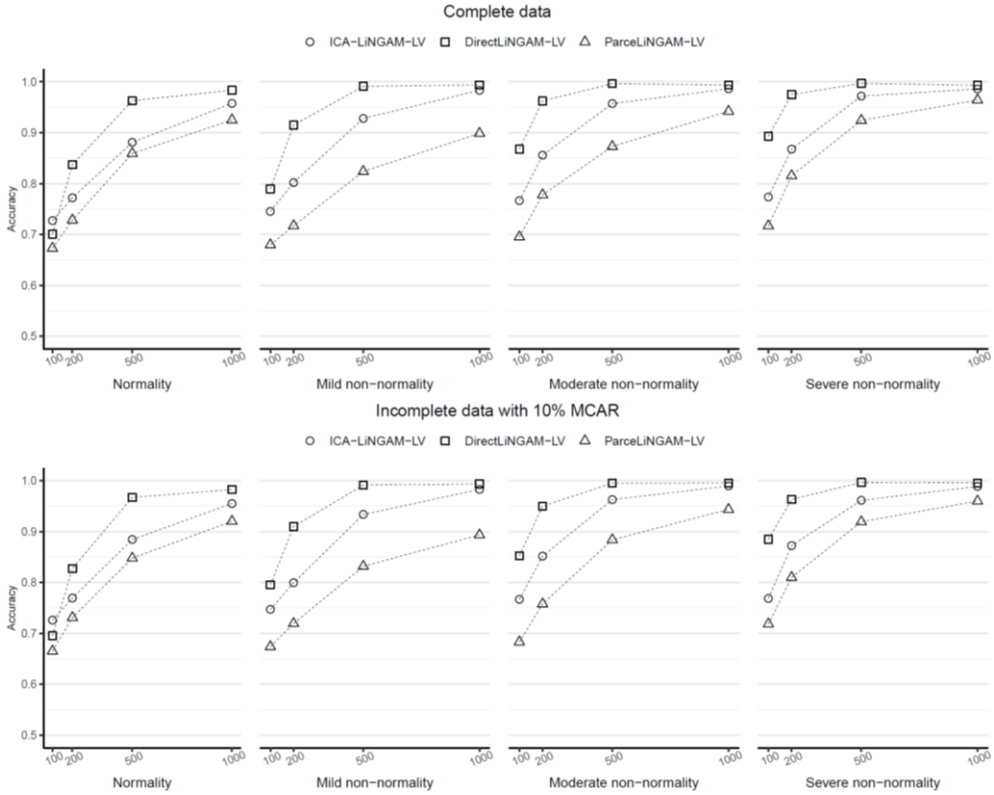
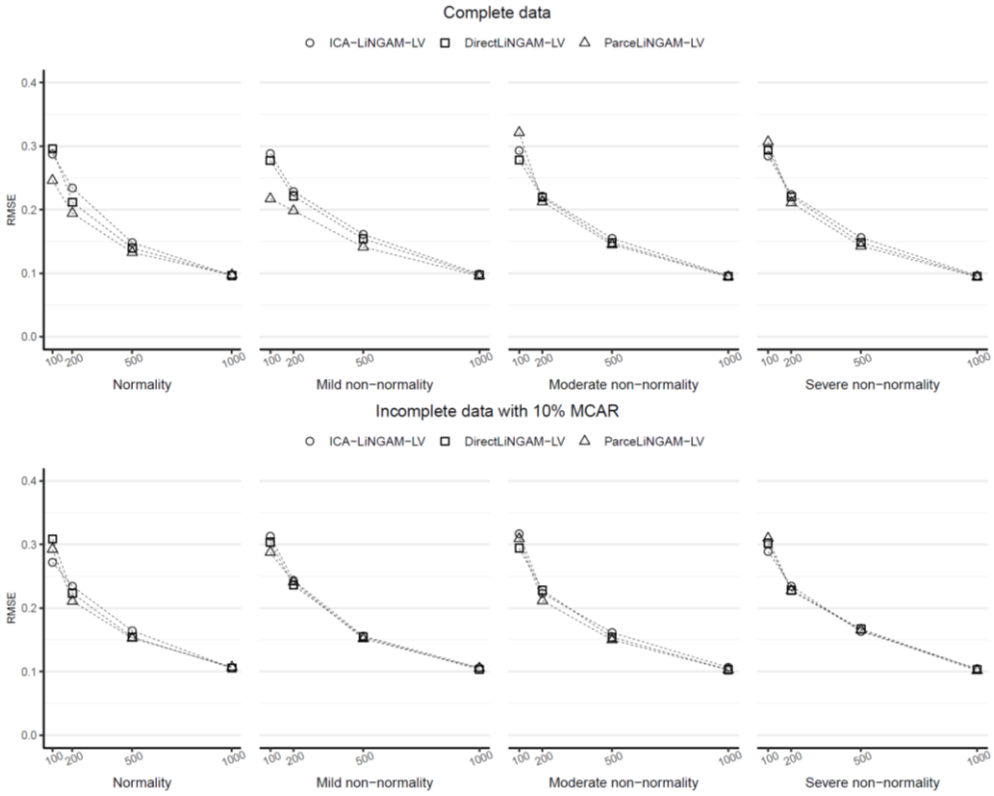


Figure 26 illustrates the mean of RMSE on PM3. The range of RMSE was from 0.094 to 0.313. With the increase in sample size, we saw a sharp decline in RMSE. The choice of the algorithm virtually had no effect on the value of RMSE.

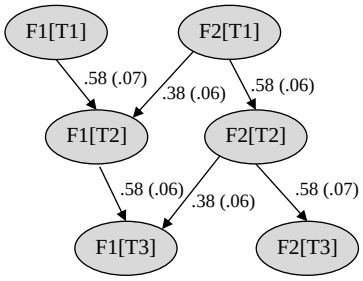
Figure 26 *Root Mean Square Error on Population Model 3*



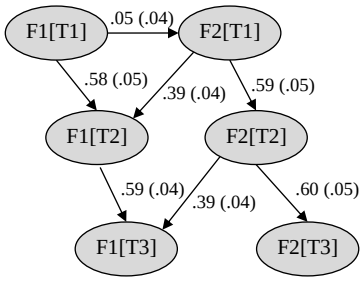
Regarding the likelihood of discovering the actual model, while DirectLiNGAM-LV had a 55.2% success rate, ICA-LiNGAM-LV (34.6%) and ParceLiNGAM-LV (24.3%). We observed two common mistakes made by the algorithms: a redundant pathway extending from F1[T1] to F2[T1] and a pathway in wrong direction from F1[T2] to F2[T1] which should have been from F2[T1] to F1[T2] (see also Figure 27).

Figure 27 *Three Most Frequently Discovered Path Models on Population Model 3*

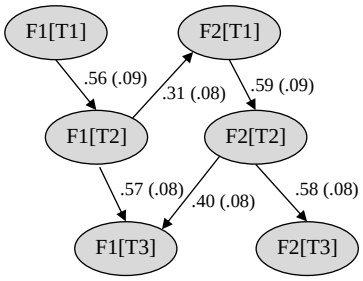
ICA-LiNGAM-LV



1st: 34.58% (*n* = 5533)
RMSEA-P: *M* = .01; *SD* = .02
CFI-P: *M* = 1.00; *SD* = .01
(Complete match)

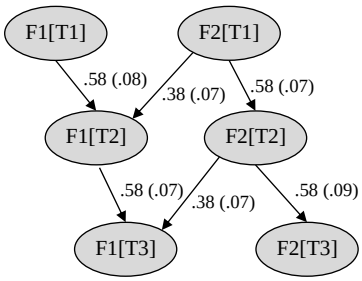


2nd: 9.42% (*n* = 1507)
RMSEA-P: *M* = .01; *SD* = .01
CFI-P: *M* = 1.00; *SD* = .00
(F1[T1]-F2[T1] pathway extra)

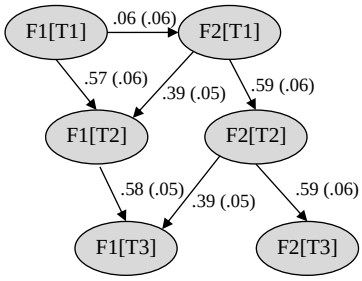


3rd: 3.94% (*n* = 631)
RMSEA-P: *M* = .01; *SD* = .02
CFI-P: *M* = 1.00; *SD* = .00
(F2[T1]-F1[T2] pathway in wrong direction)

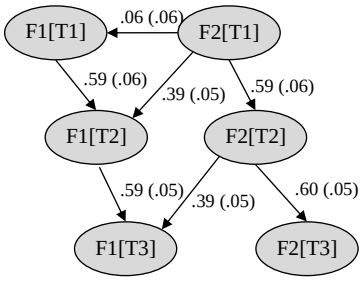
DirectLiNGAM-LV



1st: 55.23% (*n* = 8838)
RMSEA-P: *M* = .01; *SD* = .02
CFI-P: *M* = 1.00; *SD* = .01
(Complete match)

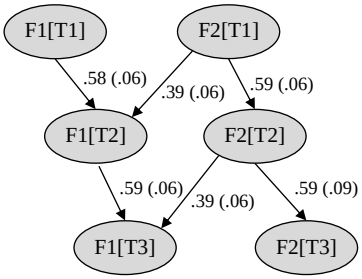


2nd: 4.45% (*n* = 712)
RMSEA-P: *M* = .01; *SD* = .01
CFI-P: *M* = 1.00; *SD* = .00
(F1[T1]-F2[T1] pathway extra)

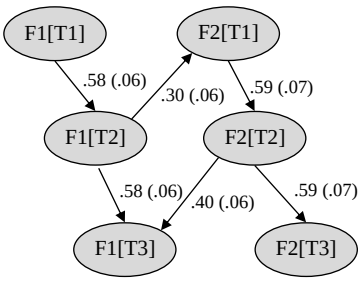


3rd: 3.23% (*n* = 517)
RMSEA-P: *M* = .01; *SD* = .01
CFI-P: *M* = 1.00; *SD* = .00
(F2[T1]-F1[T1] pathway extra)

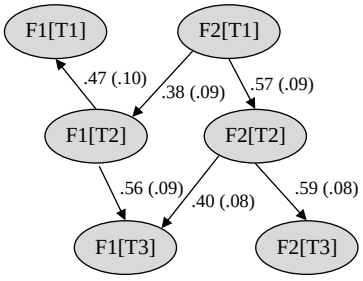
ParceLiNGAM-LV



1st: 24.29% (*n* = 3886)
RMSEA-P: *M* = .01; *SD* = .01
CFI-P: *M* = 1.00; *SD* = .00
(Complete match)



2nd: 2.69% (*n* = 430)
RMSEA-P: *M* = .06; *SD* = .02
CFI-P: *M* = .97; *SD* = .02
(F2[T1]-F1[T2] pathway in wrong direction)



3rd: 2.22% (*n* = 355)
RMSEA-P: *M* = .07; *SD* = .04
CFI-P: *M* = .97; *SD* = .03
(F1[T1]-F1[T2] pathway in wrong direction)

Chapter 6: Discussion

6.1. Study Overview

In psychology where some constructs are hidden and not directly measurable, latent-variable models are statistical methods of substantial use (Bollen, 2002), in particular SEM. In the evaluation of the fitness between models and data, SEM is vulnerable to (at least) three issues: (i) unable to pinpoint the source of misspecification, (ii) short of sensitivity to misspecification located at the path model portion, and (iii) having no information of the existence of alternative models. The consequence is that, one cannot solely reply on the model fit to justify the layout of the research model, including the directional relationships between latent variables. Instead, one is advised to consult the relevant literature and prior studies (Henley et al., 2006). However, not every empirical study is in nature confirmatory and sometimes previous studies are scant or even do not exist, under which it is difficult to make hypotheses and build a model for hypothesis testing. One solution is to turn to exploratory approaches. By far, exploratory data analysis available for causal discovery is confined to observed variables, such as PC algorithm (Spirtes & Glymour, 1991), FCI algorithm (Spirtes et al., 2000) and LiNGAM (Shimizu et al., 2006). To the best of our knowledge, no exploratory tools have been developed for causal discovery in the context of latent-variable models.

To fill the void of latent-variable causal discovery, this thesis developed a new exploratory approach called linear non-Gaussian acyclic model for latent variables (LiNGAM-LV). LiNGAM-LV is aimed at discovering causal relationships between latent variables. It requires the data and a user-specified measurement model as input. Then, LiNGAM-LV sequences the latent variables in a causal order. Afterward, it undergoes RegSEM (Jacobucci et al., 2016). The product is a discovered path model in the form of DAG. This thesis built three algorithms to implement LiNGAM-LV. The algorithms are ICA-

LiNGAM-LV, DirectLiNGAM-LV and ParceLiNGAM-LV. What makes each of the algorithms distinctive is their unique mechanism of sequencing the causal order of latent variables. In a nutshell, ICA-LiNGAM-LV uses ICA to identify LiNGAM-LV; DirectLiNGAM-LV takes advantage of mutual information; ParceLiNGAM-LV takes HSIC (Gretton et al., 2005). The R codes to implement the captioned algorithm are available in Appendix 16.

As the concept of LiNGAM-LV and its three algorithms have not been studied, this thesis conducted a Monte-Carlo simulation to examine the performance of the algorithms. In the setup, we considered two patterns of data missingness (complete data & incomplete data with 10% MCAR), four severity of data normality (from normal to severely non-normal), four sample sizes (from 100 to 1,000). To enhance the generalizability of the study finding, the simulation study covered three types of population models widely adopted in psychology (i.e., causal chain model, mediation model & cross-lagged panel model). Four performance indicators were RMSEA-P, CFI-P, accuracy and RMSE.

6.2 Summary of Study Findings

6.2.1 Sample Size

The results of our simulation study revealed that sample size was a major factor of the performance of the algorithms. That is, when the sample size increased from 100 to 1,000, the mean values of RMSEA-P and of CFI-P improved, accuracy raised and RMSE declined while controlling for the severity of non-normality, the type of population models and the type of algorithms. The finding echoed with previous studies in which sample size was positively correlated with the sensitivity of RMSEA-P (O'Boyle & Williams, 2011), the sensitivity of CFI-P (Lance et al., 2016) and accuracy (Entner & Hoyer, 2008; Shimizu et al., 2006). Also, sample size was found negatively related to RMSE (Shimizu et al., 2011; Tashiro et al., 2014). According to asymptotic statistical theory (see Bentler & Chou, 1987)

with the assumption of random sampling, when the size of a sample becomes infinitely and arbitrarily large, it reduces the discrepancy between the scores on observed variables and their population values, so does the latent variables; it leads to the improved causal sequencing of latent variables in LiNGAM and also the improved performance of simplifying the path model in RegSEM.

6.2.2 Non-normality

On average, the performance of the algorithms went up with the severity of non-normality. Precisely, non-normality affected the rate of change in the performance of the algorithms with the increase in sample size. When the non-normality level rose, the rate of change in the performance or the slope became steeper. The effect of non-normality had been documented in previous studies (e.g., Tashiro et al., 2014). To better understand the positive effect of non-normality on the performance, we revisit a core assumption of LiNGAM – non-Gaussianity. When the data distribution is deviated away from normality, it provides information from third- and higher-order moment structures that covariance matrices, equivalent to the second-order moments, cannot capture (Shimizu, 2014). The information is useful in determining the causal relationship between two variables in the sense that, when two error terms are in non-Gaussian distributions and one of them has a distribution more non-normal than the other, the distribution of the corresponding two variables differ. By comparing the distributions of the two variables, the algorithms gain insight into which is the cause and which is the effect. When the non-normality level increases, the amount of information relating to causal relationships would rise, creating a more optimal and beneficial environment for the algorithms to discover the causal. As said, sample size is also a facilitative factor of the algorithms' performance. When the severe non-normality meets up with large sample size, the performance of the algorithms would rise exponentially. Even if the non-Gaussian assumption is breached, the performance of the algorithms did not

deteriorate to a great extent, according to the study finding. It infers that the algorithms preserve some capability of handling normal data. Data missingness did not impact the performance conspicuously, however. Future studies may consider other data missing patterns (e.g., MAR & MNAR).

6.2.3 Algorithm

DirectLiNGAM-LV seemed to be the winner between three which outperformed ICA-LiNGAM-LV and ParceLiNGAM-LV in nearly every simulation conditions. In terms of the ability to discover the true model, DirectLiNGAM-LV was the highest with 89.5% in PM1, 37.2% in PM2 and 55.2% in PM3. Compared with DirectLiNGAM-LV, ICA-LiNGAM faced a drop by 1.6% in PM1, 1.3% in PM2 and 20.7% in PM3; ParceLiNGAM encountered a great fall by 17.3% in PM1, 17.9% in PM2 and 30.9% in PM3. It suggests that the lead of DirectLiNGAM-LV over the other two expanded when the number of latent variables increased from three (in PM1) to six (in PM3). It means, with more latent variables included in a path model, the advantage of using DirectLiNGAM-LV over other algorithms became more apparent. Interestingly, in a small sample size of 100 together with normal distribution, ICA-LiNGAM-LV outcompeted DirectLiNGAM-LV in terms of accuracy which applied to all population models. Under other simulation conditions, the performance of ICA-LiNGAM-LV was not that far behind that of DirectLiNGAM-LV, not as much as when compared with ParceLiNGAM-LV. ParceLiNGAM-LV was regarded as the worst performer among the three.

The relatively poor performance of PareLiNGAM-LV is likely linked to its objective. ParceLiNGAM is originally designed to handle hidden confounding variables that affects more than one known observed variables in the system. To achieve the objective, ParceLiNGAM endorses the bottom-up strategy in determining the causal order which was shown to be robust against latent confounding variables in LiNGAM (Tashiro et al., 2014).

However, in LiNGAM-LV where latent confounding variables are assumed to be accounted for by residual terms in the structural equations, the bottom-up strategy can be a liability as it increases the odd of searching a wrong parent.

6.2.4 Anomaly in ParceLiNGAM-LV

An anomaly in ParceLiNGAM-LV caught our attention. Typically, the performance of ParceLiNGAM-LV improved with sample size, no matter how deviant the data distribution was from normal. Yet, ParceLiNGAM-LV performed poorly in simulation conditions (e.g., PM1 + sample size of 100 + non-normal data). Inspecting the causal order offered us some insight into the anomaly. Given complete data with severe non-normality and at a sample size of 100, ParceLiNGAM-LV had 244 out of 500 replications (48.8%) that produced a correct order in response to PM1 (i.e., $k(F2) < k(F1) < k(F3)$). It had 50.8% chance (254 / 500) of giving an incorrect order (i.e., $k(F1) < k(F2) < k(F3)$), with a trivial 0.4% likelihood (2 / 500) of giving another incorrect order ($k(F2) < k(F3) < k(F1)$). Recall that ParceLiNGAM-LV endorses a bottom-up strategy in determining the causal order. That is, it searches the last variable at the beginning and back to the first variable. The simulation results showed that ParceLiNGAM-LV had no issue with identifying F3 as the last variable, with 498 out of 500 attempts (99.6%) successfully locating F3 as the last. But when it came to determine which was the second last variable between F1 and F2, ParceLiNGAM-LV placed roughly fifty percent chance of picking F1 as the second last variable. The same did not happen when the sample size was 500 or under, nor when the data was relatively less deviant from Gaussian distribution. For example, when sample size was 500 while keeping other conditions unchanged, ParceLiNGAM-LV yielded a correct order in 446 out of 500 attempts (89.2%). When the data non-normality condition was set to moderate while keeping other conditions intact, the algorithm produced a correct causal order in 444 of 500 attempts (88.8%). One plausible explanation to the anomaly was that, in deciding the regressor

between two factor scores where Fisher's method of combining *p*-values is not applicable, HSIC might not perform up to the expectation under the severe non-normality and small sample size. This issue may not be unique to PM1 but probably happened in other population models. Yet, the increasing number of latent variables rendered the exponential rise in the number of possible causal orders, making less noticeable the incorrect sequencing between first two latent variables in a causal order with many variables.

6.3 Contributions, Implication and Recommendation

6.3.1 Theoretical Contribution

This thesis developed LiNGAM-LV and its algorithms that enrich the variety and comprehensiveness of the exploratory data analysis, precisely the causal discovery methods. LiNGAM-LV is one of the first causal discovery methods for latent variables. Different from the traditional causal discovery methods, LiNGAM utilizes the non-Gaussian distribution of data to derive the directional relationships between observed variables and LiNGAM-LV extends to the relations between latent variables. Before the development of LiNGAM-LV, researchers have limited options to justify the causal relationships between latent variables such as prior studies. The introduction of LiNGAM-LV and its algorithms allows researchers to explore causal relationships even without prior hypotheses.

LiNGAM-LV algorithms can be treated as a follow-up of other exploratory data analysis, such as exploratory graph analysis (EGA; Golino et al., 2020). EGA is intended to determine the number of latent variables and their relationships with observed variables, without prior hypotheses. Combining EGA with LiNGAM-LV, for example, enables one to explore the number of latent variables and the directional relationship between the latent variables from scratch.

The development of LiNGAM-LV also contributes to the literature of SEM. In SEM, the integrity of the results interpretation in the research model is under the threat of the

alternative models that researchers do not consider. Yet, testing every possible alternative model can be a tedious task and the workload rises rapidly as the number of variables increases. LiNGAM-LV algorithms can produce a path model in an automatic manner which can be treated as the most representative alternative model from the data perspective. It benefits researchers greatly by saving time on searching alternative models and comparing them against the research model. With LiNGAM-LV algorithms, researchers can streamline the procedure by comparing the discovered model against the research model.

6.3.2 Practical Contributions

This thesis provides researchers with R codes to implement the algorithms which can be found in Appendix 16. A user's guide on how to use the algorithms on R is in Appendix 15. Despite the complexity of the principle behind LiNGAM-LV algorithms and the mathematical equations involved, the R codes automates the operation of the algorithms in the way that users enter the dataset, the factor loading matrix and the factor covariance matrix to the algorithms, and they will return the discovered path models within a short period of time which depends on the number of latent variables specified. The R codes given in this thesis makes LiNGAM-LV algorithms accessible to applied researchers. For those who are interested in quickly exploring the pattern of latent variables in their dataset, the R codes for LiNGAM-LV algorithms are of great assistance.

6.3.3 Implication

Amid the growing popularity of confirmatory data analysis, the development of LiNGAM-LV serves as a reminder to long forgotten statistical orientation of exploratory data analysis. Confirmatory data analysis (e.g., SEM) aims to confirm or disapprove of the hypotheses derived from theories. Due to the increased awareness of the replicability of study findings (Shrout & Rodgers, 2018) which was catalysed by a replication crisis (Open Science Collaboration, 2015), confirmatory data analysis is viewed to be a dominating research

approach. Yet, overwhelmingly emphasizing the importance of replicability comes with criticism. Baumeister (2016) argues that the over-emphasis on confirmation stifles discovery and creativity and makes empirical studies less cost-effective. Finkle et al. (2017) comment that replicability and discovery are two equally important is ways to achieve a high-quality science.

Despite the growing use of confirmatory data analysis, exploration remains crucial (Bentler, 1986). Exploratory data analysis is a toolbox operating under the philosophical framework that discovers the patterns of the variables (Fife & Rodgers, 2022). Exploratory data analysis demands no a priori hypotheses, no specification of sample sizes and no statistical plan in advance. In lieu of a binary choice, data analysis can fall into somewhere along a continuum with exploratory data analysis at one end and confirmatory data analysis at the other (Tukey, 1973). LiNGAM-LV algorithms are close to the realm of exploratory data analysis. Strictly speaking, LiNGAM-LV algorithms are exploratory data analysis that is built on the confirmatory data analysis in the way that they require pre-specified measurement models. In this regard, one can view LiNGAM-LV algorithms as a "semi-exploratory data analysis." The development of LiNGAM-LV is a response to the criticism against over-emphasis on confirmatory data analysis by raising the awareness that one can explore the latent-variable pattern with the given data. It reminds us that even though replicating previous findings is one way of doing research, discovering is another way that is equally, if not less, important.

6.3.4 Recommendation

Based on the simulation results, we advised using DirectLiNGAM-LV and ICA-LiNGAM-LV to discover path models from scratch. One should avoid ParceLiNGAM-LV for the time being. Some empirical studies (e.g., Kotoku et al., 2020) demonstrated that the path analysis models discovered by DirectLiNGAM and ICA-LiNGAM are in consensus. If

the discovered path models by the two algorithms are the same, it adds confidence to the discovered path model that it is more likely produced by the underlying causal mechanism, instead of by chance. Even though DirectLiNGAM-LV predominantly outperformed ICA-LiNGAM-LV in the simulation study, ICA-LiNGAM-LV is advantageous over DirectLiNGAM-LV in accommodating various forms of measurement models. Since ICA-LiNGAM-LV does not involve computation of factor scores, it opens up possibilities that ICA-LiNGAM-LV applies to some forms of measurement models such as bifactor models (Reise, 2012) and models with latent method factors (Horan et al., 2003).

6.4 Limitation and Future Direction

6.4.1 Causal Mechanism or by Chance

A path model discovered by the algorithms can happen either by chance, or because of the underlying causal mechanism or an interpretable process behind that reflects a causal mechanism about latent variables. As such, we cannot confidently proclaim that the discovered path model genuinely captures the causal mechanism of latent variables from the data. One way to improve the credibility of the discovered path model is to by cross-validation (Picard & Cook, 1984). In it, a number of observations were repeatedly drawn form the original dataset and fed onto the algorithms (Fife & Rodgers, 2021), to see if the product is discovered differently. Cross-validation allows for the validation of the discovered path model. The future development of LiNGAM-LV algorithms can incorporate cross-validation. Another way to eliminate the random chance from influencing the algorithm is through prior knowledge. DirectLiNGAM and ParceLiNGAM for observed variables accept prior knowledge with which they adjust the process of causal order determination (see Shimizu et al., 2011, p. 1234), while ICA-LiNGAM does not due to ICA. In prior knowledge, one can specify the relationship between two variables in three possible way : i) one variable has a direct effect on the other variable, ii) two are not related, and iii) no information is

available. Because of the scope of this thesis, we did not incorporate prior knowledge as part of DirectLiNGAM-LV and ParceLiNGAM-LV. Future studies are recommended to expand the utility of the R code for LiNGAM-LV algorithms by giving users an option to add prior knowledge such that the influence of prior knowledge on the performance of the algorithms can be evaluated.

6.4.2 Linearity

Path models discovered by the algorithm presume linear relations between latent variables. Although it is not as common, it is possible to have non-linear relationships. In SEM of latent interaction and quadratic effects (Bollen & Paxton, 1998; see also Marsh et al., 2013), some examples of modelling non-linearity include product-indicator unconstrained approach (Marsh et al., 2004) and latent moderated structural equations (Klein & Moosbrugger, 2000). In the early development of LiNGAM, Hoyer and colleagues (2009) pioneered to generalize the linear framework of LiNGAM to non-linear models. The allowance of interaction between latent variables adds complexity to the identification of LiNGAM-LV, which this thesis did not consider. Much more work is needed to integrate the non-linear LiNGAM into the latent-variable causal discovery.

6.4.3 Unidirectionality

In latent-variable models for hypothesis testing, some directional relationships are not specified as their cause-effect sequences are not hypothesized, or are not of substantive interest. Instead, the involved latent variables are allowed to covary (as indicated by bi-directional curved arcs). In LiNGAM, all pathways are restricted to be one-directional. To add flexibility to LiNGAM, one is advised to change some direct effects in the discovered model to correlation with theoretical justification.

6.4.4 Non-Reciprocity

LiNGAM does not tolerate reciprocity or feedback loop, even though reciprocity is one way to describe relationships between variables in practice. Back to the stimulation study, PM3 was designed to address the issue which itself was a cross-lagged panel model to explain reciprocal effects between two variables. At this stage, we informed LiNGAM-LV of six unrelated latent variables without mentioning the temporal relationships. The missing piece of information contributed to the reduced performance of the LiNGAM-LV algorithms. SEM with longitudinal data have several vastly studied models at its disposal (see also McArdle, 2009), including dynamic structural equation models (Asparouhov et al., 2018) and general cross-lagged panel model (Zyphur et al., 2020). In future studies, we can consider borrowing the diagram of the above models to construct prior knowledge for LiNGAM-LV algorithms and examine how the prior knowledge about temporal relationship affects the performance.

6.4.5. Estimation Method

The simulation study adopted MLR as the estimation method in RegSEM as RegSEM is not ready to handle estimators other than maximum likelihood on R (Jacobucci et al., 2021). Future studies may consider other estimation methods. While maximum likelihood estimation goes well with continuous data, other estimators are advised when dealing with non-normal data and ordered categorical data (e.g., Beauducel & Herzberg, 2006; Savalei & Rhemtulla, 2013), including diagonally weighted least squares (DWLS) and weighted least square means and variance adjusted (WLSMV) estimators. Bayesian structural equation modelling (Muthén & Asparouhov, 2012) is yet another rigorous approach to SEM that uses Bayesian analysis where information priors are part of the model estimation.

6.5 Conclusion

In the contemporary literature, latent-variable causal discovery methods are by far scarce and underdeveloped. To fill in the gap, we proposed linear non-Gaussian acyclic model for latent variables (LiNGAM-LV). It is an exploratory statistical model to identify directional relationships between latent variables. This thesis developed three algorithms to run LiNGAM-LV (i.e., ICA-LiNGAM-LV, DirectLiNGAM-LV & ParceLiNGAM-LV). The algorithms require the raw data along with the user-specified measurement model as input. Then, the algorithms sequence the latent variables in a causal order and output a path model that keeps misspecification to the minimum. Each algorithm has the unique way to sequence the causal order. The intent of the algorithms is to offer one insight into how the path model looks like from the data perspective. By running LiNGAM-LV algorithms, one can heed to the data about the pattern of the latent variables. We hope that researchers will find useful the algorithms delineated, in conjunction with R codes given in the appendices and take advantage of the discovered path models as a stimulus of exploration and creativity.

References

Anderson, J. C., & Gerbing, D. W. (1988). Structural equation modeling in practice: A review and recommended two-step approach. *Psychological Bulletin, 103*(3), 411–423. https://doi.org/10.1037/0033-2909.103.3.411

Hamaker, E. L., Kuiper, R. M., & Grasman, R. P. P. P. (2015). A critique of the cross-lagged panel model. *Psychological Methods, 20*(1), 102–116. https://doi.org/10.1037/a0038889

Rogosa, D. (1980). A critique of cross-lagged correlation. *Psychological Bulletin, 88*(2), 245–258. https://doi.org/10.1037/0033-2909.88.2.245

Shimizu, S. (2014). LiNGAM: Non-Gaussian methods for estimating causal structures. *Behaviormetrika, 41*, 65–98.

Asparouhov, T., & Muthén, B. (2009). Exploratory structural equation modeling. *Structural Equation Modeling: A Multidisciplinary Journal, 16*(3), 397–438. https://doi.org/10.1080/10705510903008204

Asparouhov, T., & Muthén, B. (2019). Nesting and equivalence testing for structural equation models. *Structural Equation Modeling: A Multidisciplinary Journal, 26*(2), 302–309. https://doi.org/10.1080/10705511.2018.1513795

Asparouhov, T., Hamaker, E. L., & Muthén, B. (2018). Dynamic structural equation models. *Structural Equation Modeling: A Multidisciplinary Journal, 25*(3), 359–388. https://doi.org/10.1080/10705511.2017.1406803

Bartlett, M. S. (1937). The statistical conception of mental factors. *British Journal of Psychology, 28*(1), 97–104. https://doi.org/10.1111/j.2044-8295.1937.tb00863.x

Baumeister, R. F. (2016). Charting the future of social psychology on stormy seas: Winners, losers, and recommendations. *Journal of Experimental Social Psychology, 66*, 153–158. https://doi.org/10.1016/j.jesp.2016.02.003

Beauducel, A., & Herzberg, P. Y. (2006). On the performance of maximum likelihood versus means and variance adjusted weighted least squares estimation in CFA. *Structural Equation Modeling, 13*(2), 186–203. https://doi.org/10.1207/s15328007sem1302_2

Bentler, P. M. (1983). Some contributions to efficient statistics in structural models: Specification and estimation of moment structures. *Psychometrika, 48*(4), 493–517. https://doi.org/10.1007/BF02293875

Bentler, P. M. (1986). Structural modeling and Psychometrika: An historical perspective on growth and achievements. *Psychometrika, 51*(1), 35–51. https://doi.org/10.1007/BF02293997

Bentler, P. M., & Chou, C. P. (1987). Practical issues in structural modeling. *Sociological Methods & Research, 16*(1), 78–117. https://doi.org/10.1177/0049124187016001004

Bentler, P. M., & Satorra, A. (2010). Testing model nesting and equivalence. *Psychological Methods, 15*(2), 111–123. https://doi.org/10.1037/a0019625

Bentler, P. M., & Yuan, K. H. (1999). Structural equation modeling with small samples: Test statistics. *Multivariate Behavioral Research, 34*(2), 181–197. https://doi.org/10.1207/S15327906Mb340203

Blanca, M. J., Arnau, J., López-Montiel, D., Bono, R., & Bendayan, R. (2013). Skewness and kurtosis in real data samples. *Methodology, 9*(2), 78–84. https://doi.org/10.1027/1614-2241/a000057

Bolkan, S., & Goodboy, A. K. (2015). Exploratory theoretical tests of the instructor humor–student learning link. *Communication Education, 64*(1), 45–64. https://doi.org/10.1080/03634523.2014.978793

Bollen, K. A. (1989). *Structural equations with latent variables.* John Wiley & Sons.

Bollen, K. A. (2002). Latent variables in psychology and the social sciences. *Annual Review of Psychology*, 53(1), 605–634. https://doi.org/10.1146/annurev.psych.53.100901.135239

Bollen, K. A., & Paxton, P. (1998). Interactions of latent variables in structural equation models. *Structural Equation Modeling: A Multidisciplinary Journal*, *5*(3), 267–293. https://doi.org/10.1080/10705519809540105

Bollen, K. A., & Pearl, J. (2013). Eight myths about causality and structural equation models. In S. L. Morgan (Ed.). *Handbook of causal analysis for social research* (pp. 301–328). Springer.

Bono, R., Blanca, M. J., Arnau, J., & Gómez-Benito, J. (2017). Non-normal distributions commonly used in health, education, and social sciences: A systematic review. *Frontiers in Psychology*, *8*, 1602. https://doi.org/10.3389/fpsyg.2017.01602

Borsboom, D., & Cramer, A. O. (2013). Network analysis: an integrative approach to the structure of psychopathology. *Annual Review of Clinical Psychology*, *9*, 91–121. https://doi.org/10.1146/annurev-clinpsy-050212-185608

Browne, M. W. (1984). Asymptotically distribution-free methods for the analysis of covariance structures. *British Journal of Mathematical and Statistical Psychology*, *37*(1), 62–83. https://doi.org/10.1111/j.2044-8317.1984.tb00789.x

Cain, M. K., Zhang, Z., & Yuan, K. H. (2017). Univariate and multivariate skewness and kurtosis for measuring nonnormality: Prevalence, influence and estimation. *Behavior Research Methods*, *49*(5), 1716–1735. https://doi.org/10.3758/s13428-016-0814-1

Chen, F., Curran, P. J., Bollen, K. A., Kirby, J., & Paxton, P. (2008). An empirical evaluation of the use of fixed cutoff points in RMSEA test statistic in structural equation models. *Sociological Methods & Research*, *36*(4), 462–494. https://doi.org/10.1177/0049124108314720

Chickering, D. M. (2002). Optimal structure identification with greedy search. *Journal of Machine Learning Research, 3*, 507–554.

Comon, P. (1994). Independent component analysis, a new concept?. *Signal Processing, 36*(3), 287–314. https://doi.org/10.1016/0165-1684(94)90029-9

Darroch, J. N., & James, I. R. (1974). F-independence and null correlation of continuous, bounded-sum, positive variables. *Journal of the Royal Statistical Society: Series B (Methodological), 36*(3), 467–483. https://doi.org/10.1111/j.2517-6161.1974.tb01023.x

Dodge, Y., & Rousson, V. (2000). Direction dependence in a regression line. *Communications in Statistics-Theory and Methods, 29*(9), 1957–1972. https://doi.org/10.1080/03610920008832589

Dodge, Y., & Rousson, V. (2001). On asymmetric properties of the correlation coefficient in the regression setting. *The American Statistician, 55*(1), 51–54.

Dray, S., & Josse, J. (2015). Principal component analysis with missing values: a comparative survey of methods. *Plant Ecology, 216*(5), 657–667. https://doi.org/10.1007/s11258-014-0406-z

Entner, D., & Hoyer, P. O. (2011). Discovering uUnconfounded cCausal rRelationships uUsing lLinear nNon-Gaussian mModels. In T. Onada, D. Bekki, & E. McCready (Eds.), *New frontiers in artificial intelligence* (pp. 181–195). Springer. https://doi.org/10.1007/978-3-642-25655-4_17

Epskamp, S., Rhemtulla, M., & Borsboom, D. (2017). Generalized network psychometrics: Combining network and latent variable models. *Psychometrika, 82*(4), 904–927. https://doi.org/10.1007/s11336-017-9557-x

Estabrook, R., & Neale, M. (2013). A comparison of factor score estimation methods in the presence of missing data: Reliability and an application to nicotine dependence.

Multivariate Behavioral Research, 48(1), 1–27. https://doi.org/10.1080/00273171.2012.730072

Fife, D. A., & Rodgers, J. L. (2022). Understanding the exploratory/confirmatory data analysis continuum: Moving beyond the "replication crisis". *American Psychologist, 77*(3), 453–466. https://doi.org/10.1037/amp0000886

Finkel, E. J., Eastwick, P. W., & Reis, H. T. (2017). Replicability and other features of a high-quality science: Toward a balanced and empirical approach. *Journal of Personality and Social Psychology, 113*(2), 244–253. https://doi.org/10.1037/pspi0000075

Fisher, R. A. (1932). *Statistical methods for research workers*. Oliver and Boyd.

Geiger, D., Verma, T., & Pearl, J. (1990). Identifying independence in Bayesian networks. *Networks, 20*(5), 507–534. https://doi.org/10.1002/net.3230200504

Gerbing, D. W., & Anderson, J. C. (1993). Monte Carlo evaluations of goodness-of-fit indices for structural equation models. In K. A. Bollen, & J. S. Long (Eds.), *Testing Structural Equation Models* (pp. 40-65). Newbury Park, CA: Sage.

Glymour, C., Zhang, K., & Spirtes, P. (2019). Review of causal discovery methods based on graphical models. *Frontiers in Genetics, 10*, 524. https://doi.org/10.3389/fgene.2019.00524

Golino, H., Shi, D., Christensen, A. P., Garrido, L. E., Nieto, M. D., Sadana, R., Thiyagarajan, J. A., & Martinez-Molina, A. (2020). Investigating the performance of exploratory graph analysis and traditional techniques to identify the number of latent factors: A simulation and tutorial. *Psychological Methods, 25*(3), 292–320. https://doi.org/10.1037/met0000255

Goretzko, D., & Bühner, M. (2020). One model to rule them all? Using machine learning algorithms to determine the number of factors in exploratory factor

analysis. *Psychological Methods, 25*(6), 776–786. https://doi.org/10.1037/met0000262

Greenland, S., Pearl, J., & Robins, J. M. (1999). Causal diagrams for epidemiologic research. *Epidemiology*, *10*(1), 37–48. https://doi.org/10.1097/00001648-199901000-00008

Gretton, A., Fukumizu, K., Teo, C. H., Song, L., Schölkopf, B., & Smola, A. J. (2007). A kernel statistical test of independence. In J. Platt, D. Koller, Y. Singer, & S. Roweis (Eds.), *Advances in Neural Information Processing Systems 20*, (pp. 585–592). MIT Press.

Gretton, A., Herbrich, R., Smola, A., Bousquet, O., & Schölkopf, B. (2005). Kernel methods for measuring independence. *Journal of Machine Learning Research, 6*, 2075–2129.

Grice, J. W. (2001). Computing and evaluating factor scores. *Psychological Methods, 6*(4), 430–450. https://doi.org/10.1037//1082-989X.6.4.430

Hayes, T., & Usami, S. (2020). Factor score regression in the presence of correlated unique factors. *Educational and Psychological Measurement, 80*(1), 5–40. https://doi.org/10.1177/0013164419854492

Henley, A. B., Shook, C. L., & Peterson, M. (2006). The presence of equivalent models in strategic management research using structural equation modeling: Assessing and addressing the problem. *Organizational Research Methods, 9*(4), 516–535. https://doi.org/10.1177/1094428106290195

Hershberger, S. L. (2006). The problem of equivalent structural models. In G. R. Hancock & R. O. Mueller (Eds.)., *Structural equation modelling: A second course* (pp. 13–41). Information Age.

Himberg, J., Hyvärinen, A., & Esposito, F. (2004). Validating the independent components of neuroimaging time series via clustering and visualization. *Neuroimage, 22*(3), 1214–1222. https://doi.org/10.1016/j.neuroimage.2004.03.027

Horan, P. M., DiStefano, C., & Motl, R. W. (2003). Wording effects in self-esteem scales: Methodological artifact or response style?. *Structural Equation Modeling, 10*(3), 435–455. https://doi.org/10.1207/S15328007SEM1003_6

Hoyer, P. O., Shimizu, S., Kerminen, A. J., & Palviainen, M. (2008). Estimation of causal effects using linear non-Gaussian causal models with hidden variables. *International Journal of Approximate Reasoning, 49*(2), 362–378. https://doi.org/10.1016/j.ijar.2008.02.006

Hu, L. T., & Bentler, P. M. (1998). Fit indices in covariance structure modeling: Sensitivity to underparameterized model misspecification. *Psychological Methods, 3*(4), 424–453. https://doi.org/10.1037/1082-989X.3.4.424

Hu, L. T., & Bentler, P. M. (1999). Cutoff criteria for fit indexes in covariance structure analysis: Conventional criteria versus new alternatives. *Structural Equation Modeling: A Multidisciplinary Journal, 6*(1), 1–55. https://doi.org/10.1080/10705519909540118

Hyvärinen, A. (1998). New approximations of differential entropy for independent component analysis and projection pursuit. In M. I. Jordan, M. J. Kearns, & S. A. Solla (Eds.), *Advances in Neural Information Processing Systems 10* (pp. 273–279). MIT Press.

Hyvärinen, A. (1999). Fast ICA for noisy data using Gaussian moments. *IEEE International Symposium on Circuits and Systems, 5*, 57–61. https://doi.org/10.1109/ISCAS.1999.777510

Hyvärinen, A., & Oja, E. (2000). Independent component analysis: algorithms and applications. *Neural Networks, 13*(4), 411–430. https://doi.org/10.1016/S0893-6080(00)00026-5

Hyvärinen, A., & Smith, S. M. (2013). Pairwise likelihood ratios for estimation of non-Gaussian structural equation models. *Journal of Machine Learning Research, 14*, 111–152.

Hyvärinen, A., Zhang, K., Shimizu, S., & Hoyer, P. O. (2010). Estimation of a structural vector autoregression model using non-gaussianity. *Journal of Machine Learning Research, 11*(5), 1709–1731.

Jacobucci, R., Grimm, K. J., & McArdle, J. J. (2016). Regularized structural equation modeling. *Structural Equation Modeling: A Multidisciplinary Journal, 23*(4), 555–566. https://doi.org/10.1080/10705511.2016.1154793

Jacobucci, R., Grimm, K. J., Brandmaier, A. M., Serang, S., Kievit, R. A., Scharf, F., Li, X.,. & Ye, A. (2021). regsem: Regularized sStructural eEquation mModeling. R package version 1.8.0. https://CRAN.R-project.org/package=regsem

James, L. R., Mulaik, S. A., & Brett, J. M. (1982). *Causal analysis: Assumptions, models, and data.* Sage.

Jöreskog, K. G. (1973). A general method for estimating a linear structural equation system. In A. Goldberger & O. D. Duncan (Eds.), *Structural equation models in the social sciences* (pp. 85–112). New York: Academic Press.

Josse, J., & Husson, F. (2012). Handling missing values in exploratory multivariate data analysis methods. *Journal de la Société Française de Statistique, 153*(2), 79–99.

Josse, J., & Husson, F. (2016). missMDA: a package for handling missing values in multivariate data analysis. *Journal of Statistical Software, 70*(1), 1–31.

Jutten, C., & Herault, J. (1991). Blind separation of sources, part I: An adaptive algorithm based on neuromimetic architecture. *Signal Processing, 24*(1), 1–10. https://doi.org/10.1016/0165-1684(91)90079-X

Klein, A., & Moosbrugger, H. (2000). Maximum likelihood estimation of latent interaction effects with the LMS method. *Psychometrika*, *65*(4), 457–474. https://doi.org/10.1007/BF02296338

Kost, J. T., & McDermott, M. P. (2002). Combining dependent p-values. *Statistics & Probability Letters, 60*(2), 183–190.

Lance, C. E., Beck, S. S., Fan, Y., & Carter, N. T. (2016). A taxonomy of path-related goodness-of-fit indices and recommended criterion values. *Psychological Methods, 21*(3), 388–404. https://doi.org/10.1037/met0000068

Lauritzen, S. L. (1996). *Graphical models.* Clarendon Press.

Liang, X., & Jacobucci, R. (2020). Regularized structural equation modeling to detect measurement bias: Evaluation of lasso, adaptive lasso, and elastic net. *Structural Equation Modeling: A Multidisciplinary Journal, 27*(5), 722–734. https://doi.org/10.1080/10705511.2019.1693273

Little, T. D. (2013). *Longitudinal structural equation modeling*. Guilford press.

Maeda, T. N., & Shimizu, S. (2020, June). RCD: Repetitive causal discovery of linear non-Gaussian acyclic models with latent confounders. *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics, 108*, pp. 735–745. Available from https://proceedings.mlr.press/v108/maeda20a.html.

Marsh, H. W., Wen, Z., & Hau, K. T. (2004). Structural equation models of latent interactions: evaluation of alternative estimation strategies and indicator construction. *Psychological Methods, 9*(3), 275–300. https://doi.org/10.1037/1082-989X.9.3.275

Marsh, H. W., Wen, Z., Hau, K. T., & Nagengast, B. (2013). Structural equation models of latent interaction and quadratic effects. In G. R. Hancock, & R. O. Mueller (Eds.), *Structural equation modelling: A second course* (pp. 267–308). Information Age Publishing.

McArdle, J. J. (2009). Latent variable modeling of differences and changes with longitudinal data. *Annual Review of Psychology, 60,* 577–605. https://doi.org/10.1146/annurev.psych.60.110707.163612

McDonald, R. P., & Ho, M. H. R. (2002). Principles and practice in reporting structural equation analyses. *Psychological Methods, 7*(1), 64–82. https://doi.org/10.1037//1082-989X.7.1.64

McDonald, R. P., & Mulaik, S. A. (1979). Determinacy of common factors: A nontechnical review. *Psychological Bulletin, 86*(2), 297–306. https://doi.org/10.1037/0033-2909.86.2.297

Micceri, T. (1989). The unicorn, the normal curve, and other improbable creatures. *Psychological Bulletin, 105*(1), 156–166. https://doi.org/10.1037/0033-2909.105.1.156

Moneta, A., Entner, D., Hoyer, P. O., & Coad, A. (2013). Causal inference by independent component analysis: Theory and applications. *Oxford Bulletin of Economics and Statistics, 75*(5), 705–730. https://doi.org/10.1111/j.1468-0084.2012.00710.x

Mooij, J., Janzing, D., Peters, J., & Schölkopf, B. (2009, June). Regression by dependence minimization and its application to causal inference in additive noise models. *Proceedings of the 26th annual international conference on machine learning*, New York, United States, pp. 745–752. https://doi.org/10.1145/1553374.1553470

Mooijaart, A. (1985). Factor analysis for non-normal variables. *Psychometrika, 50*(3), 323–342. https://doi.org/10.1007/BF02294108

Mooijaart, A., & Bentler, P. M. (2010). An alternative approach for nonlinear latent variable models. *Structural Equation Modeling, 17*(3), 357–373. https://doi.org/10.1080/10705511.2010.488997

Mooijaart, A., & Satorra, A. (2012). Moment testing for interaction terms in structural equation modeling. *Psychometrika, 77*(1), 65–84. https://doi.org/10.1007/s11336-011-9232-6

Mulaik, S. A. (2005). Looking back on the indeterminacy controversies in factor analysis. In A. Maydeu-Olivares & J. J. McArdle (Eds.), *Contemporary psychometrics: A festschrift for Roderick P. McDonald* (pp. 173–206). Lawrence Erlbaum Associates.

Mulaik, S. A. (2009). *Linear causal modeling with structural equations.* Chapman and Hall/CRC.

Mulaik, S. A., James, L. R., Van Alstine, J., Bennett, N., Lind, S., & Stilwell, C. D. (1989). Evaluation of goodness-of-fit indices for structural equation models. *Psychological Bulletin, 105*(3), 430–445. https://doi.org/10.1037/0033-2909.105.3.430

Muthén, B., & Asparouhov, T. (2012). Bayesian structural equation modeling: a more flexible representation of substantive theory. *Psychological Methods, 17*(3), 313–335. https://doi.org/10.1037/a0026802

O'Boyle, E. H., Jr., & Williams, L. J. (2011). Decomposing model fit: Measurement vs. theory in organizational research using latent variables. *Journal of Applied Psychology, 96*(1), 1–12. https://doi.org/10.1037/a0020539

Open Science Collaboration (2015). Estimating the reproducibility of psychological science. *Science*, *349*(6251), aac4716. https://doi.org/10.1126/science.aac4716

Pearl, J. (1995). Causal diagrams for empirical research. *Biometrika*, *82*(4), 669–688. https://doi.org/10.1093/biomet/82.4.669

Pearl, J. (2000). *Causality, models, reasoning, and inference.* Cambridge University Press.

Pearl, J. (2014). The causal foundations of structural equation modeling. In R. H. Hoyle (Ed.), *Handbook of structural equation modeling* (pp. 68–91), Guildford Press.

Picard, R. R., & Cook, R. D. (1984). Cross-validation of regression models. *Journal of the American Statistical Association, 79*(387), 575–583. https://doi.org/10.1080/01621459.1984.10478083

Pollaris, A., & Bontempi, G. (2020). Latent causation: An algorithm for pairs of correlated latent variables in linear non-Gaussian structural equation modeling. *BNAIC/BeneLearn 2020*, 209.

Qu, W., & Zhang, Z. (2020). mnonr: A generator of multivariate non-normal random numbers. R package version 1.0.0. https://CRAN.R-project.org/package=mnonr

Qu, W., Liu, H., & Zhang, Z. (2020). A method of generating multivariate non-normal random numbers with desired multivariate skewness and kurtosis. *Behavior Research Methods, 52*(3), 939–946. https://doi.org/10.3758/s13428-019-01291-5

R Core Team (2021). R: A Language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria.

Raykov, T., & Marcoulides, G. A. (2001). Can there be infinitely many models equivalent to a given covariance structure model?. *Structural Equation Modeling, 8*(1), 142–149. https://doi.org/10.1207/S15328007SEM0801_8

Raykov, T., & Penev, S. (1999). On structural equation model equivalence. *Multivariate Behavioral Research, 34*(2), 199–244. https://doi.org/10.1207/S15327906Mb340204

Reise, S. P. (2012). The rediscovery of bifactor measurement models. *Multivariate Behavioral Research, 47*(5), 667–696. https://doi.org/10.1080/00273171.2012.715555

Robins, J. M., Scheines, R., Spirtes, P., & Wasserman, L. (2003). Uniform consistency in causal inference. *Biometrika, 90*(3), 491–515. https://doi.org/10.1093/biomet/90.3.491

Rosseel, Y. (2012). Lavaan: An R package for structural equation modeling and more. *Journal of Statistical Software, 48*(2), 1–36.

Roy, S. N. (1958). Step-down procedure in multivariate analysis. *The Annals of Mathematical Statistics, 29*(4), 1177–1187. https://doi.org/10.1214/aoms/1177706449

Savalei, V., & Rhemtulla, M. (2013). The performance of robust test statistics with categorical data. *British Journal of Mathematical and Statistical Psychology, 66*(2), 201–223. https://doi.org/10.1111/j.2044-8317.2012.02049.x

Schouten, R. M., Lugtig, P., & Vink, G. (2018). Generating missing values for simulation purposes: a multivariate amputation procedure. *Journal of Statistical Computation and Simulation, 88*(15), 2909–2930. https://doi.org/10.1080/00949655.2018.1491577

Shimizu, S., & Kano, Y. (2008). Use of non-normality in structural equation modeling: Application to direction of causation. *Journal of Statistical Planning and Inference, 138*(11), 3483–3491.

Shimizu, S., Hoyer, P. O., & Hyvärinen, A. (2009). Estimation of linear non-Gaussian acyclic models for latent factors. *Neurocomputing, 72*(7), 2024–2027. https://doi.org/10.1016/j.neucom.2008.11.018

Shimizu, S., Hoyer, P. O., Hyvärinen, A., Kerminen, A., & Jordan, M. (2006). A linear non-Gaussian acyclic model for causal discovery. *Journal of Machine Learning Research, 7*(10), 2003–2030.

Shimizu, S., Inazumi, T., Sogawa, Y., Hyvärinen, A., Kawahara, Y., Washio, T., Hoyer, P. O., & Bollen, K. (2011). DirectLiNGAM: A direct method for learning a linear non-Gaussian structural equation model. *Journal of Machine Learning Research, 12*, 1225–1248.

Spirtes, P., & Glymour, C. (1991). An algorithm for fast recovery of sparse causal graphs. *Social Science Computer Review, 9*(1), 62–72. https://doi.org/10.1177/089443939100900106

Spirtes, P., Glymour, C. N., Scheines, R., & Heckerman, D. (2000). *Causation, prediction, and search.* MIT press.

Stelzl, I. (1986). Changing a causal hypothesis without changing the fit: Some rules for generating equivalent path models. *Multivariate Behavioral Research, 21*(3), 309–331. https://doi.org/10.1207/s15327906mbr2103_3

Székely, G. J., Rizzo, M. L., & Bakirov, N. K. (2007). Measuring and testing dependence by correlation of distances. *The Annals of Statistics, 35*(6), 2769–2794. https://doi.org/10.1214/009053607000000505

Takahashi, Y., Ozaki, K., Roberts, B. W., & Ando, J. (2012). Can low bBehavioral Activation activation sSystem predict depressive mood?: An application of non-normal structural equation modeling. *Japanese Psychological Research, 54*(2), 170–181. https://doi.org/10.1111/j.1468-5884.2011.00492.x

Tashiro, T., Shimizu, S., Hyvärinen, A., & Washio, T. (2014). ParceLiNGAM: Aa causal ordering method robust against latent confounders. *Neural Computation, 26*(1), 57–83. https://doi.org/10.1162/NECO_a_00533

Ten Berge, J. M., Krijnen, W. P., Wansbeek, T., & Shapiro, A. (1999). Some new results on correlation-preserving factor scores prediction methods. *Linear Algebra and its Applications, 289*(1), 311–318. https://doi.org/10.1016/S0024-3795(97)10007-6

Thurstone, L. L. (1937). Ability, motivation, and speed. *Psychometrika, 2*(4), 249–254. https://doi.org/10.1007/BF02287896

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological), 58*(1), 267–288. https://doi.org/10.1111/j.1467-9868.2011.00771.x

Van Buuren, S., & Groothuis-Oudshoorn, K. (2011). mice: Multivariate imputation by chained equations in R. *Journal of Statistical Software, 45*(1), 1–67.

VanderWeele, T. J., & Robins, J. M. (2007). Directed acyclic graphs, sufficient causes, and the properties of conditioning on a common effect. *American Journal of Epidemiology, 166*(9), 1096–1104. https://doi.org/10.1093/aje/kwm179

von Eye, A., & DeShon, R. P. (2012). Directional dependence in developmental research. *International Journal of Behavioral Development, 36*(4), 303–312. https://doi.org/10.1177/0165025412439968

von Eye, A., & Wiedermann, W. (2014). On direction of dependence in latent variable contexts. *Educational and Psychological Measurement, 74*(1), 5–30. https://doi.org/10.1177/0013164413505863

Wei, M., Heppner, P. P., & Mallinckrodt, B. (2003). Perceived coping as a mediator between attachment and psychological distress: A structural equation modeling approach. *Journal of Counseling Psychology, 50*(4), 438–447. https://doi.org/10.1037/0022-0167.50.4.438

Weinberger, N. (2018). Faithfulness, coordination and causal coincidences. *Erkenntnis, 83*(2), 113–133. https://doi.org/10.1007/s10670-017-9882-6

Wiedermann, W., & Li, X. (2018). Direction dependence analysis: A framework to test the direction of effects in linear models with an implementation in SPSS. *Behavior Research Methods, 50*(4), 1581–1601. https://doi.org/10.3758/s13428-018-1031-x

Wiedermann, W., & Sebastian, J. (2020). Direction dependence analysis in the presence of confounders: Applications to linear mediation models using observational data. *Multivariate Behavioral Research, 55*(4), 495–515. https://doi.org/10.1080/00273171.2018.1528542

Wiedermann, W., & von Eye, A. (2015). Direction of effects in multiple linear regression models. *Multivariate Behavioral Research, 50*(1), 23–40. https://doi.org/10.1080/00273171.2014.958429

Williams, L. J., & Holahan, P. J. (1994). Parsimony-based fit indices for multiple-indicator models: Do they work?. *Structural Equation Modeling: A Multidisciplinary Journal, 1*(2), 161–189. https://doi.org/10.1080/10705519409539970

Williams, L. J., & O'Boyle Jr, E. (2011). The myth of global fit indices and alternatives for assessing latent variable relations. *Organizational Research Methods, 14*(2), 350–369. https://doi.org/10.1177/1094428110391472

Yuan, K. H. (2005). Fit indices versus test statistics. *Multivariate Behavioral Research, 40*(1), 115–148. https://doi.org/10.1207/s15327906mbr4001_5

Yuan, K. H., & Bentler, P. M. (1998). Normal theory based test statistics in structural equation modelling. *British Journal of Mathematical and Statistical Psychology, 51*(2), 289–309. https://doi.org/10.1111/j.2044-8317.1998.tb00682.x

Yuan, K. H., & Bentler, P. M. (2000). Three likelihood-based methods for mean and covariance structure analysis with nonnormal missing data. *Sociological Methodology, 30*(1), 165–200. https://doi.org/10.1111/0081-1750.00078

Zhang, J., & Spirtes, P. (2016). The three faces of faithfulness. *Synthese, 193*(4), 1011–1027. https://doi.org/10.1007/s11229-015-0673-9

Zou, H. (2006). The adaptive lasso and its oracle properties. *Journal of the American Statistical Association, 101*(476), 1418–1429. https://doi.org/10.1198/016214506000000735

Zyphur, M. J., Allison, P. D., Tay, L., Voelkle, M. C., Preacher, K. J., Zhang, Z., Hamaker, E. L., Shamsollahi, A., Pierides, D. C., Koval, P., & Diener, E. (2020). From data to causes I: Building a general cross-lagged panel model (GCLM). *Organizational Research Methods, 23*(4), 651–687. https://doi.org/10.1177/1094428119847278

Appendices

Appendix 1: Operational Definition of Structural Coefficients

Pearl (2009) addresses that the operational definition of regression coefficients (denoted by γ) in linear regression models is not the same as that of structural coefficients (denoted by β) in structural equation models. Consider a linear regression model between two observed variables, say x_i and x_j, that:

$$x_i = \gamma_{ij} x_j + e_i^{(j)} \tag{A1.1}$$

The interpretation of γ_{ij} is:

$$\gamma_{ij} = \mathrm{E}\left[x_i \middle| x_j = J + 1\right] - \mathrm{E}\left[x_i \middle| x_j = J\right] \tag{A1.2}$$

Literally, γ_{ij} is the change in the conditional expected value of x_i by one-unit increase in x_j from value J to $J + 1$. It does not enforce any cause-effect concept onto x_i and x_j. A structural equation between two latent variables, say η_i and η_j, can be specified as:

$$\eta_i = \beta_{ij} \eta_j + \varepsilon_i^{(j)} \tag{A1.3}$$

The meaning of β_{ij} is:

$$\beta_{ij} = \mathrm{E}\left[\eta_i \middle| \mathrm{do}\left(\eta_j = J + 1\right)\right] - \mathrm{E}\left[\eta_i \middle| \mathrm{do}\left(\eta_j = J\right)\right] \tag{A1.4}$$

where "do($\eta_j = J$)" is an operator about a deliberate action or an external force of holding η_j at level J. In words, β_{ij} is the change in the controlled expectation (not conditional expectation) of η_i given that the magnitude of the external force applying onto η_j increases from value J to $J + 1$. It treats the change in η_j as a hypothetical intervention, whatever it is gathered from an experimental study or an observational study. It enforces the idea that η_j exerts a causal effect on η_i provided that the set of causal assumptions behinds is acceptable (Pearl, 2009, p. 161).

Appendix 2: Sequence of Nested Models

A2.1 Definition of Nested Models

To better understand the development of path-related fit indices, this appendix will revisit the concept of nested models to describe the relationship between two models, say Models A and B. Nesting is defined in two ways (Mulaik et al., 1989). One is the covariance matrix nesting (Bentler & Bonett, 1980) by which the nested relationship means the set of covariance matrices of Model A is a subset of the set of covariance matrices of Model B. The other is the parameter nesting (Bentler & Bonett, 1980) by which the nested relationship means the model parameter vectors of Model A is identical to that of Model B except for some parameter being constrained to equalities or known values (e.g., zeros) in Model A. All in all, covariance matrix nesting is a less restrictive definition of nesting while parameter nesting is a more restrictive definition (Mulaik et al., 1989). For simplicity's sake, we use nested in a less restrictive sense; that is, nested always means covariance matrix-nested, unless specified otherwise.

Consider that Model A is nested within Model B which in turn nested within a third model called Model C. Models A, B and C form a sequence of nested models, the idea that is believed to be first advocated by Roy (1958). A sequence of nested models (or the hierarchy of models) is the result of ordering multiple models by covariance-matrix nesting. According to Mulaik et al. (1989), the sequence of nested model goes from a saturated model to the measurement model, and stop at a null model. At one extreme, a saturated model refers to the most constrained model that allows observed variables to correlate with each other and thus the least constraint, resulting in a perfect fit to the observed covariance matrix of the data. At the other extreme, a null (or independence) model refers to the least constrained model that observed variables are constrained from correlating. The sequence of nested models have been applied to the development of fit indices.

A2.2 Hierarchy of Nested Models

With the number of proposed fit indices growing, scholars started to categorize and name models along the sequence of nested models. Williams and Holahan (1994) review the proposed fit indices at the time and give labels to some models along the sequence of nested models to facilitate communication. They distinguish two types of saturated models. A *completely saturated model* (CS) is the most constrained model that allows all observed variables to correlate. A *structural saturated model* (SS) is a somewhat constrained model that allows all possible pathways connecting latent variables to estimate. SS shares the same degrees of freedom with the measurement model. The difference between the two is that, a mix of directional pathways and correlations between latent variables are present in SS whilst only latent variable correlations are present in the measurement model. In the middle of the sequence is a *theoretical model* (T), the hypothetical latent-variable model that users propose for hypothesis testing. T is said to be parameter-nested within SS, and T is covariance matrix-nested within the measurement model. A *structural null model* (SN) is a relatively free model nested within T where all possible pathways are fixed to zero, with latent exogenous variables allowed to covary with each other. An *absolute null model* (AN) is the least constrained model where observed variables are uncorrelated. See Williams and O'Boyle's (2011)'s paper for the discussion on the sequence of nested models.

From the simulation study design that follows, we use PM1 to illustrate the sequence of nested models. Given that PM1 has 15 observed endogenous variables (*x*1 to *x*15), one latent exogenous variable (i.e., F2) and two latent endogenous variables (F1 & F3), the expression in matrix form is (see Mulaik, 2009):

$$\begin{bmatrix}\boldsymbol{\eta}\\ \mathbf{x}\end{bmatrix} = \mathbf{A}\begin{bmatrix}\boldsymbol{\eta}\\ \mathbf{x}\end{bmatrix} + \boldsymbol{\Gamma}\begin{bmatrix}\boldsymbol{\xi}\\ \mathbf{y}\end{bmatrix} + \begin{bmatrix}\boldsymbol{\Delta} & \mathbf{0}\\ \mathbf{0} & \boldsymbol{\Psi}\end{bmatrix}\begin{bmatrix}\boldsymbol{\zeta}\\ \boldsymbol{\varepsilon}\end{bmatrix} \tag{A2.1}$$

where $\boldsymbol{\eta}$ is a ($q_2 \times 1$) random vector of latent endogenous variables, $\mathbf{x}$ is a ($p_2 \times 1$) random vector of observed endogenous variables, $\boldsymbol{\xi}$ is a ($q_1 \times 1$) random vector of latent exogenous

variables, $\mathbf{y}$ is a ($p_1 \times 1$) random vector of observed exogenous variables, $\boldsymbol{\zeta}$ is a ($q_2 \times 1$) random vector of disturbances on latent endogenous variables, $\boldsymbol{\varepsilon}$ is a ($p_2 \times 1$) random vector of disturbances on observed endogenous variables. (The total number of observed variables is $p_1 + p_2 = p$, and the total number of latent variables is $q_1 + q_2 = q$.) $\mathbf{A}$ is a [($q_2 + p_2$) × ($q_2 + p_2$)] matrix of structural coefficients that relate endogenous variables to other endogenous variables, $\boldsymbol{\Gamma}$ is a [($q_2 + p_2$) × ($q_1 + p_1$)] matrix of structural coefficients that relate endogenous variables to other exogenous variables. $\boldsymbol{\Delta}$ is a ($q_2 \times q_2$) diagonal matrix of structural coefficients that relate latent endogenous variables to the corresponding disturbances denoted by $\boldsymbol{\zeta}$, $\boldsymbol{\Psi}$ is a ($p_2 \times p_2$) diagonal matrix of structural coefficients that relate observed endogenous variables to the corresponding disturbances denoted by $\boldsymbol{\varepsilon}$. Typically, the connection strength of the structural coefficients linking endogenous variables to their disturbances is set to one which makes $\boldsymbol{\Delta}$ and $\boldsymbol{\Psi}$ an identity matrix. Please note that we use a different set of notations here for the illustration of the sequence of nested models where the exogenous variables are distinguished from the endogenous variables. Yet, in the main text, we stick to the expressions in equations 1.1 and 1.2 which do not require the differentiation between latent endogenous and latent exogenous variables as such information is not given in the causal discovery of latent variables.

Using PM1 as an illustration of the hierarchy of nested models, we set $p_1 = 0$, $p_2 = 15$, $q_1 = 1$ and $q_2 = 2$, leading to $\mathbf{y} = \emptyset$, $\mathbf{x} = (x_1, \ldots, x_{15})^T$, $\boldsymbol{\eta} = (\text{F1}, \text{F3})^T$, $\boldsymbol{\xi} = (\text{F2})^T$, $\boldsymbol{\zeta} = (d_{\text{F1}}, d_{\text{F3}})^T$ and $\boldsymbol{\varepsilon} = (e_1, \ldots, e_{15})^T$. The expression of T with respect to PM1 in matrix form is:

$$\begin{bmatrix} \boldsymbol{\eta} \\ \mathbf{x} \end{bmatrix} = \begin{bmatrix} \mathbf{B}_{\boldsymbol{\eta}\boldsymbol{\eta}} & \mathbf{0} \\ \boldsymbol{\Lambda}_{\mathbf{x}\boldsymbol{\eta}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{\eta} \\ \mathbf{x} \end{bmatrix} + \begin{bmatrix} \mathbf{B}_{\boldsymbol{\eta}\boldsymbol{\xi}} & \mathbf{0} \\ \boldsymbol{\Lambda}_{\mathbf{x}\boldsymbol{\xi}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{\xi} \\ \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{\zeta} \\ \boldsymbol{\varepsilon} \end{bmatrix}$$

where $\mathbf{B}_{\boldsymbol{\eta}\boldsymbol{\eta}}$ is a ($q_2 \times q_2$) matrix of structural coefficients between latent endogenous variables, $\mathbf{B}_{\boldsymbol{\eta}\boldsymbol{\xi}}$ is a ($q_2 \times q_1$) matrix of structural coefficients connecting latent endogenous variables to latent exogenous variables, $\boldsymbol{\Lambda}_{\mathbf{x}\boldsymbol{\eta}}$ is a ($p_2 \times q_2$) matrix of factor loadings connecting observed

endogenous variables to latent endogenous variables, $\mathbf{\Lambda_{x\xi}}$ is a ($p_2 \times q_1$) matrix of factor loadings connecting observed endogenous variables to latent exogenous variables.

$$\mathbf{B_{\eta\eta}} = \begin{bmatrix} 0 & 0 \\ b_{\mathrm{F3,F1}} & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0.60 & 0 \end{bmatrix}, \mathbf{B_{\eta\xi}} = \begin{bmatrix} b_{\mathrm{F1,F2}} \\ 0 \end{bmatrix} = \begin{bmatrix} 0.60 \\ 0 \end{bmatrix},$$

$$\mathbf{\Lambda_{x\eta}} = \begin{bmatrix} \lambda_{x1,F1} & 0 \\ \lambda_{x2,\mathrm{F1}} & 0 \\ \lambda_{x3,\mathrm{F1}} & 0 \\ \lambda_{x4,\mathrm{F1}} & 0 \\ \lambda_{x5,\mathrm{F1}} & \lambda_{x5,\mathrm{F3}} \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \lambda_{x11,\mathrm{F1}} & \lambda_{x11,\mathrm{F3}} \\ 0 & \lambda_{x12,\mathrm{F3}} \\ 0 & \lambda_{x13,\mathrm{F3}} \\ 0 & \lambda_{x14,\mathrm{F3}} \\ 0 & \lambda_{x15,\mathrm{F3}} \end{bmatrix} = \begin{bmatrix} 0.70 & 0 \\ 0.70 & 0 \\ 0.70 & 0 \\ 0.70 & 0 \\ 0.70 & 0.21 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0.21 & 0.70 \\ 0 & 0.70 \\ 0 & 0.70 \\ 0 & 0.70 \\ 0 & 0.70 \end{bmatrix}, \mathbf{\Lambda_{x\xi}} = \begin{bmatrix} \lambda_{x1,\mathrm{F2}} \\ 0 \\ 0 \\ 0 \\ 0 \\ \lambda_{x6,\mathrm{F2}} \\ \lambda_{x7,\mathrm{F2}} \\ \lambda_{x8,\mathrm{F2}} \\ \lambda_{x9,\mathrm{F2}} \\ \lambda_{x10,\mathrm{F2}} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.21 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0.70 \\ 0.70 \\ 0.70 \\ 0.70 \\ 0.70 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix},$$

$$\mathrm{Var}(\boldsymbol{\xi}) = \begin{bmatrix} \xi_{\mathrm{F2,F2}} \end{bmatrix} = [1.00], \mathrm{Var}(\boldsymbol{\zeta}) = \begin{bmatrix} \zeta_{\mathrm{F1,F1}} & 0 \\ 0 & \zeta_{\mathrm{F3,F3}} \end{bmatrix} = \begin{bmatrix} 0.64 & 0 \\ 0 & 0.64 \end{bmatrix} \text{ and}$$

$$\mathrm{diag}[\mathrm{Var}(\boldsymbol{\varepsilon})] = \left(\varepsilon_{x1,x1}, \varepsilon_{x2,x2}, \varepsilon_{x3,x3}, \varepsilon_{x4,x4}, \dots, \varepsilon_{x15,x15}\right)^{\mathrm{T}} =$$

$$(0.2895, 0.51, 0.51, 0.51, \dots, 0.51)^{\mathrm{T}}$$

SS with respect to PM1 is the same as that of T, except that all possible pathways are allowed to freely estimate. In the expression of SS, almost all matrices are duplicates from those of T but:

$$\mathbf{B_{\eta\xi}} = \begin{bmatrix} b_{\mathrm{F1,F2}} \\ b_{\mathrm{F3,F2}} \end{bmatrix}$$

SN with respect to PM1 is close to that of T, except that all possible pathways have their estimates fixed to zero. In the expression of SN, all matrices look the same as those of T except for:

$$\mathbf{B_{\eta\eta}} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} = \mathbf{0}$$

In CS with respect to PM1, all latent variables are absent (i.e., $q_1 = q_2 = 0$, $\boldsymbol{\eta} = \emptyset, \boldsymbol{\xi} = \emptyset$) and all observed variables are allowed to covary. Thereby, several matrices become null:

$$\mathbf{B}_{\boldsymbol{\eta}\boldsymbol{\eta}} = \mathbf{0}, \mathbf{B}_{\boldsymbol{\eta}\boldsymbol{\xi}} = \mathbf{0}, \boldsymbol{\Lambda}_{\mathbf{x}\boldsymbol{\eta}} = \mathbf{0}. \boldsymbol{\Lambda}_{\mathbf{x}\boldsymbol{\xi}} = \mathbf{0}$$

As a result, the expression of CS can be simplified to $\mathbf{x} = \mathbf{I}\boldsymbol{\varepsilon}$ where var($\boldsymbol{\varepsilon}$) is a non-diagonal ($p_2 \times p_2$) matrix filled with non-zero elements throughout, not just along the diagonal. In AN with respect to PM1, all latent variables are absent and all observed variables are not permitted to covary. It leads to the expression of AN as $\mathbf{x} = \mathbf{I}\boldsymbol{\varepsilon}$ where diag[var($\boldsymbol{\varepsilon}$)]=$\left(\varepsilon_{x1,x1},\varepsilon_{x2,x2},,\ldots,\varepsilon_{x15,x15}\right)^{T}$. If one arranges the five models from the sequence of nested models by the number of free parameters, we obtain CS > SS > T > SN > AN.

Appendix 3: Solutions to Permutation and Scaling Indeterminacies

Let $\mathbf{W}_{\text{ica}}$ be a ($p' \times p$) squared mixing matrix estimated from ICA, where the length of p' is equal to the length of p,. Yet, the sequence of variables in p' does not correspond to that of p and thus $\mathbf{W}_{\text{ica}}$ is not identified. A solution to the permutation indeterminacy is to take advantage of the properties of DAG by permuting $\mathbf{W}_{\text{ica}}$ to a form in which all its diagonal elements are of zero values. The respective permutation on $\mathbf{W}_{\text{ica}}$ is expressed as:

$$\bar{\mathbf{P}}\mathbf{W}_{\text{ica}} = \bar{\mathbf{P}}\mathbf{P}\widetilde{\mathbf{W}}_{\text{ica}} \tag{A3.1}$$

where $\bar{\mathbf{P}}$ is a ($p \times p'$) permutation matrix; $\mathbf{P}$ is another ($p' \times p$) permutation matrix, the same as in equation 2.7. The chosen $\bar{\mathbf{P}}$ is able to neutralize $\mathbf{P}$ (i.e., $\bar{\mathbf{P}}\mathbf{P} = \mathbf{I}$). $\widetilde{\mathbf{W}}_{\text{ica}}$ is a ($p \times p$) unmixing matrix which has a correct order correspondence between its row and its column (Note that its two dimensions, the two ps, are without apostrophe). In other words, $\widetilde{\mathbf{W}}_{\text{ica}}$ is free of permutation indeterminacy.

Resolving Scaling Indeterminacy. Due to scaling indeterminacy that the variances in error terms (i.e., var(**s**)) are unknown, ICA fixes the variances itself to unity (Shimizu et al., 2011). Let $\mathbf{D}$ be a ($p \times p$) scaling matrix that contains the diagonal elements of $\widetilde{\mathbf{W}}_{\text{ica}}$ such that:

$$\mathbf{D} = \text{diag}(\widetilde{\mathbf{W}}_{\text{ica}})$$

If one divides $\widetilde{\mathbf{W}}_{\text{ica}}$ by $\mathbf{D}$, it gives:

$$\frac{\widetilde{\mathbf{W}}_{\text{ica}}}{\mathbf{D}} = \frac{\widetilde{\mathbf{W}}_{\text{ica}}}{\text{diag}\left(\widetilde{\mathbf{W}}_{\text{ica}}\right)} = \widetilde{\mathbf{W}}'_{\text{ica}} \tag{A3.2}$$

It means that standardization is applied to every row of $\widetilde{\mathbf{W}}_{\text{ica}}$ and thus yields $\widetilde{\mathbf{W}}'_{\text{ica}}$ whose diagonal elements are at numerical value ones. By setting variances in independent components to unity, scaling indeterminacy is resolved. Combining equation A3.1 with A3.2, one obtains:

$$\mathbf{W}_{\text{ica}} = \mathbf{P}\mathbf{D}\widetilde{\mathbf{W}}'_{\text{ica}}$$

which gives equation 2.7.

Appendix 4: Approximation for Matrix Permutation to Strict Lower Triangularity

Suppose $\mathbf{\Gamma}_{\text{ica}}$ be a ($p \times p$) matrix obtained from ICA that is not in strict lower triangularity. Shimizu et al. (2011)'s method is to find out a permuted $\mathbf{\Gamma}_{\text{ica}}$ in the form of strict lower triangularity through the iterative removal of elements in $\mathbf{\Gamma}_{\text{ica}}$ and continual testing $\mathbf{\Gamma}_{\text{ica}}$ for strict lower triangularity. It can save time during the permutation on $\mathbf{\Gamma}_{\text{ica}}$, in particular those of large dimensions. The proposed method begins with setting the $p(p + 1)/2$ smallest elements in absolute values in $\mathbf{\Gamma}_{\text{ica}}$ to zero. Using the four-variable case (x_{a}, x_{b}, x_{c}, x_{d}) as an example, whose true causal diagram is shown in Figure 7a, let us say the respective $\mathbf{\Gamma}_{\text{ica}}$ be:

$$\mathbf{\Gamma}_{\text{ica}} = \begin{pmatrix} \gamma'_{\text{aa}} & \gamma'_{\text{ab}} & \gamma'_{\text{ac}} & \gamma'_{\text{ad}} \\ \gamma'_{\text{ba}} & \gamma'_{\text{bb}} & \gamma'_{\text{bc}} & \gamma'_{\text{bd}} \\ \gamma'_{\text{ca}} & \gamma'_{\text{cb}} & \gamma'_{\text{cc}} & \gamma'_{\text{cd}} \\ \gamma'_{\text{da}} & \gamma'_{\text{db}} & \gamma'_{\text{dc}} & \gamma'_{\text{dd}} \end{pmatrix} = \begin{pmatrix} 0 & 1.2 & 2.2 & 1.1 \\ 0.8 & 0 & 0.3 & 2.1 \\ 0.1 & 1.3 & 0 & 1.5 \\ 0.7 & 0.2 & 0.5 & 0 \end{pmatrix}$$

In forcing 10 (= 4 (4 + 1) / 2) smallest elements in absolute values in $\mathbf{\Gamma}_{\text{ica}}$ to zero, we set six qualified elements to zero values as the four diagonal elements in the original $\mathbf{\Gamma}_{\text{ica}}$ are already at zero values:

$$\mathbf{\Gamma}_{\text{ica}} = \begin{pmatrix} \gamma'_{\text{aa}} & \gamma'_{\text{ab}} & \gamma'_{\text{ac}} & \gamma'_{\text{ad}} \\ \gamma'_{\text{ba}} & \gamma'_{\text{bb}} & \gamma'_{\text{bc}} & \gamma'_{\text{bd}} \\ \gamma'_{\text{ca}} & \gamma'_{\text{cb}} & \gamma'_{\text{cc}} & \gamma'_{\text{cd}} \\ \gamma'_{\text{da}} & \gamma'_{\text{db}} & \gamma'_{\text{dc}} & \gamma'_{\text{dd}} \end{pmatrix} = \begin{pmatrix} 0 & 1.2 & 2.2 & 1.1 \\ 0 & 0 & 0 & 2.1 \\ 0 & 1.3 & 0 & 1.5 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Assume an empty list as K and define a $\mathbf{\Gamma}'_{\text{ica}}$ which is equal to $\mathbf{\Gamma}_{\text{ica}}$ (i.e., $\mathbf{\Gamma}'_{\text{ica}} \coloneqq \mathbf{\Gamma}_{\text{ica}}$). For any row with zero-value elements in $\mathbf{\Gamma}'_{\text{ica}}$, the corresponding variable is the first variable in the causal order for it has no parent. Its index is loaded onto the tail of K and its row and column in $\mathbf{\Gamma}'_{\text{ica}}$ be removed. Back to the four-variable case, since the 4$^{\text{th}}$ and last row of $\mathbf{\Gamma}'_{\text{ica}}$ has elements all at zero values, the corresponding variable, x_{d}, is then determined as the first variable; its subscript, d, enter the tail of K such that K now is:

$$K\text{: } k(\text{d})$$

Once the causal order of x_{d} is identified, $\mathbf{\Gamma}'_{\text{ica}}$ shrinks to:

$$\mathbf{\Gamma}'_{\text{ica}} = \begin{pmatrix} \gamma'_{\text{aa}} & \gamma'_{\text{ab}} & \gamma'_{\text{ac}} \\ \gamma'_{\text{ba}} & \gamma'_{\text{bb}} & \gamma'_{\text{bc}} \\ \gamma'_{\text{ca}} & \gamma'_{\text{cb}} & \gamma'_{\text{cc}} \end{pmatrix} = \begin{pmatrix} 0 & 1.2 & 2.2 \\ 0 & 0 & 0 \\ 0 & 1.3 & 0 \end{pmatrix}$$

The test continues searching the next variable. Should it succeed, the index of the next variable will enter the end of K and the dimensions of $\mathbf{\Gamma}'_{\text{ica}}$ reduce further by one**.** If the test fails to search any row with zero-value elements at any point, it adjusts $\mathbf{\Gamma}_{\text{ica}}$ by setting its next smallest elements in absolute value to zero. Then the test resumes by letting $K \coloneqq \emptyset$ and $\mathbf{\Gamma}'_{\text{ica}} \coloneqq \mathbf{\Gamma}_{\text{ica}}$. The process repeats until one element remains in $\mathbf{\Gamma}'_{\text{ica}}$ and K has $(p - 1)$ indices arranged in a causal order. With the index of the remaining variable in $\mathbf{\Gamma}'_{\text{ica}}$ added to its tail, K is completed as the causal order of variables that shows which variables are ancestors and which are descendants. In the four-variable example, the second row of $\mathbf{\Gamma}'_{\text{ica}}$ has all elements at zero values so the corresponding variable, x_{b}, is the second variable whose variable subscript, b, enters the tail of K. K is updated to:

$$\text{K: } k(\text{d}) < k(\text{b})$$

And the $\mathbf{\Gamma}'_{\text{ica}}$ further shrinks by one more dimension to:

$$\mathbf{\Gamma}'_{\text{ica}} = \begin{pmatrix} \gamma'_{\text{aa}} & \gamma'_{\text{ab}} \\ \gamma'_{\text{ca}} & \gamma'_{\text{cb}} \end{pmatrix} = \begin{pmatrix} 0 & 2.2 \\ 0 & 0 \end{pmatrix}$$

Repeating the process, we find x_{c} to be the third variable in the causal order which updates K to:

$$K\text{: } k(\text{d}) < k(\text{b}) < k(\text{c})$$

The search pauses as K has three variable subscripts at the moment (d, b and c). The remaining variable subscript, a, enters the end of K to complete K:

$$K\text{: } k(\text{d}) < k(\text{b}) < k(\text{c}) < k(\text{a})$$

Given $k(\text{d}) < k(\text{a})$, we can only conclude that x_{d} is an ancestor of x_{a} but are not certain if x_{d} has a direct edge onto x_{a}.

Appendix 5: Prove of Mutual Information as A Function of One-dimensional Entropies

This appendix is to approve how mutual information can be expressed with one-dimensional differential entropies (Hyvärinen, 1998). Assume two observed variables, $x_i^{'}$ and $x_j^{'}$. Between the two, we run two regressions and obtain two error terms, $e_i^{(j)}$ for $x_i^{'}$ and $e_j^{(i)}$ for $x_j^{'}$. For $x_i^{'}$, the mutual information (MI) between the regressor $x_j^{'}$ and the error term $e_i^{(j)}$ is:

$$\mathrm{MI}\left(x_j^{'},e_i^{(j)}\right) = H\left(x_j^{'}\right) + H\left(e_i^{(j)}\right) - H\left(x_j^{'},e_i^{(j)}\right) \tag{A5.1}$$

For $x_j^{'}$, the MI between the regressor $x_i^{'}$ and the error term $e_j^{(i)}$ is:

$$\mathrm{MI}\left(x_i^{'},e_j^{(i)}\right) = H\left(x_i^{'}\right) + H\left(e_j^{(i)}\right) - H\left(x_i^{'},e_j^{(i)}\right) \tag{A5.2}$$

where $H(u)$ denotes the one-dimensional differential entropies of variable u, and $H(u, v)$ denotes the two-dimensional differential entropies of variables u and v. Since $H(u, v)$ is more computationally demanding in comparison to $H(u)$, it is preferable to avoid $H(u, v)$ in the calculation of MI. Hyvärinen and Smith (2013) suggest a solution with an equality that:

$$H\left(x_j^{'},e_i^{(j)}\right) = H\left(x_i^{'},e_j^{(i)}\right) \tag{A5.3}$$

To show that the equality in A5.3 holds, consider first a linear transformation from $x_j^{'}$ and $e_i^{(j)}$ to variables u and v:

$$H(u, v) = H\left(x_j^{'},e_i^{(j)}\right) + \log|\det \mathbf{A}| \tag{A5.4}$$

where $\begin{pmatrix} u \\ v \end{pmatrix} = \mathbf{A}\begin{pmatrix} x_j^{'} \\ e_i^{(j)} \end{pmatrix}$. The relation that $\begin{pmatrix} x_j^{'} \\ x_i^{'} \end{pmatrix} = \mathbf{A}\begin{pmatrix} x_j^{'} \\ e_i^{(j)} \end{pmatrix}$ holds when $\mathbf{A}$ is set to $\begin{pmatrix} 1 & 0 \\ \gamma_{ij} & 1 \end{pmatrix}$.

Substituting $x_j^{'}$, $x_i^{'}$ and $\begin{pmatrix} 1 & 0 \\ \gamma_{ij} & 1 \end{pmatrix}$ for u, v and $\mathbf{A}$, respectively, one has:

$$H\left(x_j', x_i'\right) = H\left(x_j', e_i^{(j)}\right) + \log\left|\det\begin{pmatrix} 1 & 0 \\ \gamma_{ij} & 1 \end{pmatrix}\right| = H\left(x_j', e_i^{(j)}\right) + \log 1 = H\left(x_j', e_i^{(j)}\right)$$

By the same principle,

$$H\left(x_i', x_j'\right) = H\left(x_i', e_j^{(i)}\right) + \log\left|\det\begin{pmatrix} 1 & 0 \\ \gamma_{ji} & 1 \end{pmatrix}\right| = H\left(x_i', e_j^{(i)}\right) + \log 1 = H\left(x_i', e_j^{(i)}\right)$$

Given the symmetric property of the two-dimensional differential entropy that:

$$H(u,v) = H(v,u) \tag{A5.5}$$

It is clear that:

$$H\left(x_j', e_i^{(j)}\right) = H\left(x_j', x_i'\right) = H\left(x_i', x_j'\right) = H\left(x_i', e_j^{(i)}\right)$$

With equation A5.3, the difference in MI between equations A5.1 and A5.2 (denoted by ΔMI) is:

$$\begin{aligned}\Delta\text{MI} &= \text{MI}\left(x_j', e_i^{(j)}\right) - \text{MI}\left(x_i', e_j^{(i)}\right) \\ &= H\left(x_j'\right) + H\left(e_i^{(j)}\right) - H\left(x_j', e_i^{(j)}\right) - H\left(x_i'\right) - H\left(e_j^{(i)}\right) + H\left(x_i', e_j^{(i)}\right) \\ &= H\left(x_j'\right) + H\left(e_i^{(j)}\right) - H\left(x_i'\right) - H\left(e_j^{(i)}\right)\end{aligned}$$

which resembles equation 2.12. This prove is also applicable to factor scores, simply by replacing observed variables (e.g., x_i') with factor scores (e.g., f_i').

Appendix 6: Algorithm of ICA-LiNGAM

Input: A ($N \times p$) data matrix as $\mathbf{X}$ where N is the number of observations and p is the number of observed variables.

1. Given a ($p \times 1$) random vector of observed variables as $\mathbf{x}$, define $U = \{1, \ldots p\}$ as a set of observed-variable subscripts. $|U|$ denotes the length of U.
2. Subtract the means from observed variables in $\mathbf{x}$. Apply a fast ICA algorithm that uses hyperbolic tangent function to decompose p observed variables into p independent components**.** From the solution of the fast ICA algorithm, retrieve a ($p \times p$) mixing matrix as $\mathbf{A}_{\text{ica}}$, the inverse of which is a ($p \times p$) unmixing matrix as $\mathbf{W}_{\text{ica}}$ where $\mathbf{W}_{\text{ica}}=\mathbf{A}_{\text{ica}}^{-1}$.
3. Find the unique permutation of rows of $\mathbf{W}_{\text{ica}}$ that seeks to minimizes $\sum_{\text{i}} \frac{1}{|\widetilde{\mathbf{W}}_{\text{ii}}|}$ such that the permuted matrix as $\widetilde{\mathbf{W}}_{\text{ica}}$ has non-zero elements along its diagonal.
4. Divide each row of $\widetilde{\mathbf{W}}_{\text{ica}}$ by the corresponding diagonal element to obtain $\widetilde{\mathbf{W}}'_{\text{ica}}$ (i.e., $\widetilde{\mathbf{W}}'_{\text{ica}}=\widetilde{\mathbf{W}}_{\text{ica}}/\text{diag}(\widetilde{\mathbf{W}}_{\text{ica}})$).
5. Subtract one from all diagonal elements in $\widetilde{\mathbf{W}}'_{\text{ica}}$ to yield $\mathbf{\Gamma}_{\text{ica}}$ (i.e., $\mathbf{\Gamma}_{\text{ica}}=\mathbf{I}-\widetilde{\mathbf{W}}'_{\text{ica}}$).
6. Initialize an empty list as $K \coloneqq \emptyset$ that represents the causal order $k(i)$ of the observed variables in $\mathbf{x}$, where $i = 1, 2, \ldots, p$. To determine K, find the unique permutation matrix as $\widetilde{\mathbf{P}}$ of $\mathbf{\Gamma}_{\text{ica}}$ that yields a matrix as $\widetilde{\mathbf{\Gamma}}_{\text{ica}}=\widetilde{\mathbf{P}}\mathbf{\Gamma}_{\mathbf{ica}}\widetilde{\mathbf{P}}^{\text{T}}$**.** The resultant $\widetilde{\mathbf{\Gamma}}_{\text{ica}}$ comes close to a strictly lower triangular structure as much as possible. In practice, to speed up the process of the matrix permutation to strictly lower triangularity, Shimizu et al. (2011)'s approximate method is used. Set the $p(p + 1)/2$ smallest elements in absolute value of $\mathbf{\Gamma}_{\text{ica}}$ to zero. Repeat
    a. Test if $\mathbf{\Gamma}_{\text{ica}}$ is permutable to be in strictly lower triangularity. If yes, break and return the permuted $\mathbf{\Gamma}_{\text{ica}}$ which is $\widetilde{\mathbf{\Gamma}}_{\text{ica}}$.

b. Set the additional smallest element in absolute value of $\mathbf{\Gamma}_{\text{ica}}$ to zero.

7. Substantiate a ($p \times p$) null matrix as $\mathbf{\Gamma}$. Let $U' \coloneqq U$. Repeat

   a. Define x_i where $k(i) < k(j)$ for all $j \in U'$ ($i \neq j$).

   b. Gather $\mathbf{x}_h = \{x_h\}$ where $k(h) < k(i)$ for all $h \in U$ ($i \neq h$). Should $\mathbf{x}_h$ be empty, skip to Step 7d.

   c. Run an adaptive lasso regression of x_i on $\mathbf{x}_h$. Gather a vector of regression coefficients for $\mathbf{x}_h$ to the row corresponding to x_i in $\mathbf{\Gamma}$.

   d. Let $U' \coloneqq U' \setminus i$. If $|U'| = 0$, break and return $\mathbf{\Gamma}$.

Output: A ($p \times p$) regression coefficient matrix $\mathbf{\Gamma}$.

Appendix 7: Algorithm of DirectLiNGAM

Input: A ($N \times p$) data matrix as **X** where N is the number of observations and p is the number of observed variables.

1. Given a ($p \times 1$) random vector of observed variables as **x**, define $U = \{1, \ldots p\}$ as a set of observed-variable subscripts. $|U|$ denotes the length of U. Initialize an empty list as $K := \emptyset$ that represents the causal order $k(i)$ of the observed variables in **x**, where $i = 1, 2, \ldots, p$. Let $\mathbf{x}' := \mathbf{x}$ and $U' := U$.
2. Repeat for all $i \in U'$
   a. Perform the least squares regression of x_i' on x_j' for all $j \in U'$ ($i \neq j$) and retrieve the error as $e_i^{(j)}$. Compute the independence between the regressor and the error in terms of the mutual information (MI) which is a function of one-dimensional differential entropies (H):

$$\mathrm{MI}\left(x_j', e_i^{(j)}\right) = H(x_j') + H\left(e_i^{(j)}\right) - H\left(x_j', e_i^{(j)}\right)$$

   b. Perform the least squares regression of x_j' on x_i' for all $j \in U'$ ($i \neq j$) and retrieve the error as $e_j^{(i)}$. Compute the independence between the regressor and the error:

$$\mathrm{MI}\left(x_i', e_j^{(i)}\right) = H(x_i') + H\left(e_j^{(i)}\right) - H\left(x_i', e_j^{(i)}\right)$$

   c. Subtract $\mathrm{MI}\left(x_i', e_j^{(i)}\right)$ from $\mathrm{MI}\left(x_j', e_i^{(j)}\right)$ to yield $\Delta\mathrm{MI}$:

$$\Delta\mathrm{MI} = \mathrm{MI}\left(x_j', e_i^{(j)}\right) - \mathrm{MI}\left(x_i', e_j^{(i)}\right).$$

   d. Compute the M criterion for x_i' by aggregating all ΔMIs for x_i' against every x_j' for all $j \in U'$ ($i \neq j$):

$$M(x_i') = -\sum_{j \neq i, j=1}^{p'} \min(0, \Delta\mathrm{MI})^2$$

e. Find an observed variable with the maximum value of M criterion in $\mathbf{x}'$ and denote it as x'_m. Add the corresponding variable subscript, m, to the tail of K.

f. Perform the least squares regressions of x'_i on x'_m for all $i \in U'$ ($i \neq m$) and retrieve a vector of the error terms as $\mathbf{r}^{(m)}$. Update $\mathbf{x}' \coloneqq \mathbf{r}^{(m)}$ and $U' \coloneqq U' \backslash m$. If $|U'|=1$, break and add the remaining variable subscript to the tail of K.

3. Substantiate a ($p \times p$) null matrix as $\mathbf{\Gamma}$. Reset $U' \coloneqq U$. Repeat:

   a. Define x_i where $k(i) < k(j)$ for all $j \in U'$ ($i \neq j$).

   b. Gather $\mathbf{x}_h = \{x_h\}$ where $k(h) < k(i)$ for all $h \in U$ ($i \neq h$). Should $\mathbf{x}_h$ be empty, jump to Step 3d.

   c. Run an adaptive lasso regression of x_i on $\mathbf{x}_h$. Gather a vector of regression coefficients for $\mathbf{x}_h$ to the row corresponding to x_i in $\mathbf{\Gamma}$.

   d. Update $U' \coloneqq U' \backslash i$. If $|U'| = 0$, break and return $\mathbf{\Gamma}$.

Output: A ($p \times p$) regression coefficient matrix as $\mathbf{\Gamma}$

Appendix 8: Algorithm of ParceLiNGAM

Input: A ($N \times p$) data matrix as $\mathbf{X}$ where N is the number of observations and p is the number of observed variables.

1. Given a ($p \times 1$) random vector of observed variables as $\mathbf{x}$, define $U = \{1, \ldots p\}$ as a set of observed-variable subscripts in $\mathbf{x}$. $|U|$ denotes the length of U. Initialize an empty list as $K \coloneqq \emptyset$ that represents the causal order $k(i)$ of the observed variables in $\mathbf{x}$, where $i = 1, 2, \ldots, p$. Let $\mathbf{x}' \coloneqq \mathbf{x}$ and $U' \coloneqq U$.
2. Repeat for all $i \in U'$
    a. Gather every x_j' for all $j \in U'$ ($i \neq j$) as $\mathbf{x}_{(-i)}'$. Run the least square regression of x_i' on $\mathbf{x}_{(-i)}'$ and retrieve the residual as $r_i^{(-i)}$.
    b. Compute the Hilbert-Schmidt independence criterion (HSIC) test statistic for every x_j' within $\mathbf{x}_{(-i)}'$ and $r_i^{(-i)}$ as $\chi^2_{\text{HSIC}}\left(x_j', r_i^{(-i)}\right)$.
    c. Gather all HSIC test statistics using the Fisher's combination method, where $p_{\text{HSIC}}\left(x_j', r_i^{(-i)}\right)$ is the p-value of $\chi^2_{\text{HSIC}}\left(x_j', r_i^{(-i)}\right)$

$$\chi^2_{\text{HSIC+Fisher}}\left(\mathbf{x}_{(-i)}', r_i^{(-i)}\right) = -2\Sigma_{i \neq j} \log\left\{p_{\text{HSIC}}\left(x_j', r_i^{(-i)}\right)\right\}$$

    d. Find a x_m' whose $\mathbf{x}_{(-m)}'$ and $r_m^{(-m)}$ are independent the most by:

$$x_m' = \arg\max_{i \in U \backslash K} p_{\text{HSIC+Fisher}}\left(\mathbf{x}_{(-i)}', r_i^{(-i)}\right)$$

    where $p_{\text{HSIC+Fisher}}$ is the p-value of $\chi^2_{\text{HSIC+Fisher}}$.
    e. Add the subscript m to the top of K. Update $\mathbf{x}' \coloneqq \mathbf{x}_{(-m)}'$ and $U' \coloneqq U \backslash m$. If $|U'| = 1$, break and return K.
3. Substantiate a ($p \times p$) null matrix as $\mathbf{\Gamma}$. Reset $U' \coloneqq U$. Repeat:
    a. Define x_i where $k(i) < k(j)$ for all $j \in U'$ ($i \neq j$).

b. Gather $\mathbf{x}_h=\{x_h\}$ where $k(h) < k(i)$ for all $h \in U$ $(i \neq h)$. Should $\mathbf{x}_h$ be empty, jump to Step 3d.

c. Run an adaptive lasso regression of x_i on $\mathbf{x}_h$. Gather a vector of regression coefficients for $\mathbf{x}_h$ to the row corresponding to x_i in $\mathbf{\Gamma}$.

d. Update $U' \coloneqq U' \backslash i$. If $|U'| = 0$, break and return $\mathbf{\Gamma}$.

Output: A $(p \times p)$ regression coefficient matrix as $\mathbf{\Gamma}$

Appendix 9: Algorithm of ICA-LiNGAM-LV

Input: A ($N \times p$) data matrix as $\mathbf{X}$, a ($p \times q$) factor loading matrix as $\mathbf{\Lambda}$, where N is the number of observations, p is the number of observed variables and q is the number of latent variables.

A designated number of iterations in independent component analysis (ICA) as N_{ica}.

1. Given a ($p \times 1$) vector of observed variables as $\mathbf{x}$ and a ($q \times 1$) vector of latent variables as $\boldsymbol{\eta}$, define $U = \{1,\ldots, q\}$ as a set of latent-variable subscripts in $\boldsymbol{\eta}$. $|U|$ denotes the length of U.
2. Repeat N_{ica} times ($n = 1, \ldots, N_{\text{ica}}$)
    a. Apply a fast ICA algorithm along with hyperbolic tangent function, taking $\mathbf{X}$ as a data input and specifying q as the number of independent components, which generates a ($p \times q$) mixing matrix as $\mathbf{A}_{\text{ica}}$. Compute a ($q \times q$) unmixing matrix as $\mathbf{W}_{\text{ica}}$ with the relation that $\mathbf{W}_{\text{ica}}=\left(\mathbf{\Lambda}^{\mathrm{T}}\mathbf{A}_{\text{ica}}\right)^{-1}\left(\mathbf{\Lambda}^{\mathrm{T}}\mathbf{\Lambda}\right)$.
    b. Find the unique permutation of rows of $\mathbf{W}_{\text{ica}}$ that seeks to minimizes $\sum_i \frac{1}{|\widetilde{W}_{ii}|}$ such that the permuted matrix as $\widetilde{\mathbf{W}}_{\text{ica}}$ has non-zero elements along its diagonal.
    c. Divide each row of $\widetilde{\mathbf{W}}_{\text{ica}}$ by the corresponding diagonal element to obtain $\widetilde{\mathbf{W}}'_{\text{ica}}$ (i.e., $\widetilde{\mathbf{W}}'_{\text{ica}}=\widetilde{\mathbf{W}}_{\text{ica}}/\text{diag}\left(\widetilde{\mathbf{W}}_{\text{ica}}\right)$).
    d. Subtract one from all diagonal elements in $\widetilde{\mathbf{W}}'_{\text{ica}}$ to yield $\mathbf{B}_{\text{ica}}$ (i.e., $\mathbf{B}_{\text{ica}}=\mathbf{I} - \widetilde{\mathbf{W}}'_{\text{ica}}$).
    e. Initialize an empty list as $K^{(n)} := \emptyset$ that represent the causal order $k(i)$ of the latent variables in $\boldsymbol{\eta}$, where $i = 1, 2, \ldots, q$. To determine $K^{(n)}$, find the unique permutation matrix as $\widetilde{\mathbf{P}}$ of $\mathbf{B}_{\text{ica}}$ that yields a matrix as $\widetilde{\mathbf{B}}_{\text{ica}}=\widetilde{\mathbf{P}}\mathbf{B}_{\text{ica}}\widetilde{\mathbf{P}}^{\mathrm{T}}$. The resultant $\widetilde{\mathbf{B}}_{\text{ica}}$ comes close to a strictly lower triangular structure as much as

possible. To speed up the process of the matrix permutation to strictly lower triangularity, Shimzu et al. (2011)'s approximate method is used. Set the $q(q + 1)/2$ smallest elements in absolute value in $\mathbf{B}_{\text{ica}}$ to zero. Repeat

i. Test if $\mathbf{B}_{\text{ica}}$ is permutable to be in strictly lower triangularity. If yes, break and return the permuted $\mathbf{B}_{\text{ica}}$ which is $\widetilde{\mathbf{B}}_{\text{ica}}$.

ii. Set the next smallest elements in absolute value of $\mathbf{B}_{\text{ica}}$ to zero.

3. Gather all $K^{(n)}$ for all $n = 1, \ldots, N_{\text{ica}}$, from which select a $K^{(n)}$ with the most frequency and rename it to K. If a tie happens, go back to Step 2 and run an additional N_{ica} time.
4. Initialize a $(q \times q)$ null matrix as $\mathbf{B}$. Let $U' \coloneqq U$. Repeat
   a. Define η_i where $k(i) < k(j)$ for all $j \in U'$ $(i \neq j)$.
   b. Gather $\boldsymbol{\eta}_{(h)} = \{\eta_h\}$ where $k(h) < k(i)$ for all $h \in U$ $(i \neq h)$. Should $\boldsymbol{\eta}_{(h)}$ be empty, skip to Step 4d.
   c. Run a regularized structural equation modelling of η_i on $\boldsymbol{\eta}_{(h)}$. Gather a vector of structural parameters on $\boldsymbol{\eta}_{(h)}$t o the row corresponding to η_i in $\mathbf{B}$.
   d. Update $U' \coloneqq U' \backslash i$. If $|U'| = 0$, break and return $\mathbf{B}$.
5. Run a regularized structural equation modelling of $\boldsymbol{\eta}$, where the directional relations between latent variables are specified in accordance with $\mathbf{B}$. The respective measurement model copies the one that generated $\boldsymbol{\Lambda}$. Retrieve a $(q \times q)$ structural parameter matrix as $\widetilde{\mathbf{B}}$.

Output: A $(q \times q)$ structural parameter matrix $\widetilde{\mathbf{B}}$

Appendix 10: Algorithm of DirectLiNGAM-LV

Input: A **(**$N \times p$) data matrix as **X**, a ($p \times q$) factor loading matrix as **Λ** and a ($q \times q$) factor covariance matrix as **Φ**, where N is the number of observations, p is the number of observed variables and q is the number of latent variables.

1. Given a ($p \times 1$) vector of observed variables as **x** and a ($q \times 1$) vector of latent variables as **η**, define $U = \{1,\ldots, q\}$ as a set of latent-variable subscripts in **η**. $|U|$ denotes the length of U. Initialize an empty list as $K \coloneqq \emptyset$ that represents the causal order $k(i)$ of the latent variables in **η**, where $i = 1, 2, \ldots, q$.
2. Compute the estimated scores on latent variables, the equivalent of factor scores as $\mathbf{f}=\left(f_1,\ldots,f_q\right)^{\mathrm{T}}$ based on **X**, **Λ** and **Φ**. Let $\mathbf{f}' \coloneqq \mathbf{f}$ and $U' \coloneqq U$. Repeat for all $i \in U'$
    a. Run the least squares regression of f_i' on f_j' for all $i, j \in U'$ ($i \neq j$) and retrieve the error as $e_i^{(j)}$. Compute the independence between the regressor and the error with the mutual information (MI) which is a function of one-dimensional differential entropies (H):
    $$\mathrm{MI}\left(f_j', e_i^{(j)}\right) = H\left(f_j'\right) + H\left(e_i^{(j)}\right) - H\left(f_j', e_i^{(j)}\right)$$
    b. Run the least squares regression of f_j' on f_i' for all $i, j \in U'$ ($i \neq j$) and retrieve the error as $e_j^{(i)}$. Compute the independence between the regressor and the error with MI:
    $$\mathrm{MI}\left(f_i', e_j^{(i)}\right) = H\left(f_i'\right) + H\left(e_j^{(i)}\right) - H\left(f_i', e_j^{(i)}\right)$$
    c. Subtract $\mathrm{MI}\left(f_i', e_j^{(i)}\right)$ from $\mathrm{MI}\left(f_j', e_i^{(j)}\right)$ to yield the difference in MIs as ΔMI:
    $$\Delta\mathrm{MI} = \mathrm{MI}\left(f_j', e_i^{(j)}\right) - \mathrm{MI}\left(f_i', e_j^{(i)}\right)$$
    d. Compute the M criterion for f_i' that aggregates all ΔMIs for f_i' against every f_j' for all $j \in U'$ ($i \neq j$):

$$\mathrm{M}(f_i')=-\sum_{j\neq i,\, j=1}^{p}\min\left(0,\,\Delta\mathrm{MI}\left[f_i',f_j'\right]\right)^2$$

e. Find a latent variable with the maximum value of *M* criterion in **f'** and denote it as f_m'. add the corresponding latent-variable subscript, *m*, to the tail of *K*.

f. Perform least squares regressions of f_i' on f_m' for all $i \in U'$ ($i \neq m$) and retrieve the vector of residuals as $\mathbf{r}^{(\mathrm{m})}$. Update $\mathbf{f}' \coloneqq \mathbf{r}^{(\mathrm{m})}$ and $U' \coloneqq U' \backslash m$. If $|U'| = 1$, break and add the remaining latent-variable subscript to the tail of *K*.

3. Initialize a ($q \times q$) null matrix as **B**. Reset $U' \coloneqq U$. Repeat

   a. Define η_i where $k(i) < k(j)$ for all $j \in U'$ ($i \neq j$).

   b. Gather $\boldsymbol{\eta}_{(h)} = \{\eta_h\}$ where $k(h) < k(i)$ for all $h \in U$ ($i \neq h$). Should $\boldsymbol{\eta}_{(h)}$ be empty, skip to Step 3d.

   c. Run a regularized structural equation modelling of η_i on $\boldsymbol{\eta}_{(h)}$. Gather a vector of structural parameters on $\boldsymbol{\eta}_{(h)}$ to the row corresponding to η_i in **B**.

   d. Update $U' \coloneqq U' \backslash i$. If $|U'| = 0$, break and return **B**.

4. Run a regularized structural equation modelling of **η**, where the directional relations between latent variables are specified in accordance with **B**. The respective measurement model copies the one that generated **Λ** and **Φ**. Retrieve a ($q \times q$) structural parameter matrix as $\widetilde{\mathbf{B}}$.

Output: A ($q \times q$) structural parameter matrix as $\widetilde{\mathbf{B}}$

Appendix 11: Algorithm of ParceLiNGAM-LV

Input: A ($N \times p$) data matrix as $\mathbf{X}$, a ($p \times q$) factor loading matrix as $\mathbf{\Lambda}$ and a ($q \times q$) factor covariance matrix as $\mathbf{\Phi}$, where N is the number of observations, p is the number of observed variables and q is the number of latent variables.

1. Given a ($p \times 1$) vector of observed variables as $\mathbf{x}$ and a ($q \times 1$) vector of latent variables as $\boldsymbol{\eta}$, define $U = \{1,\ldots, q\}$ as a set of latent-variable subscripts in $\boldsymbol{\eta}$. $|U|$ denotes the length of U. Initialize an empty list as $K \coloneqq \emptyset$ that represents the causal order $k(i)$ of the latent variables in $\boldsymbol{\eta}$, where $i = 1, 2, \ldots, q$.
2. Compute the estimated scores on latent variables, the equivalent of factor scores as $\mathbf{f} = \left(f_1,\ldots,f_q\right)^{\mathrm{T}}$ based on $\mathbf{X}$, $\mathbf{\Lambda}$ and $\mathbf{\Phi}$. Let $\mathbf{f}' \coloneqq \mathbf{f}$ and $U' \coloneqq U$. Repeat for all $i \in U'$:
    a. Gather every f_j' for all $j \in U'$ ($i \neq j$) as $\mathbf{f}_{(-i)}'$. Run a least square regression of f_i' on $\mathbf{f}_{(-i)}'$ and retrieve the error term as $r_i^{(-i)}$.
    b. Compute the Hilbert-Schmidt independence criterion (HSIC) test statistic for every f_j' within $\mathbf{f}_{(-i)}'$ and $r_i^{(-i)}$ as $\chi^2_{\mathrm{HSIC}}\left(f_j', r_i^{(-i)}\right)$.
    c. Gather all HSIC test statistics using the Fisher's combination method, where $p_{\mathrm{HSIC}}\left(f_j', r_i^{(-i)}\right)$ is the p-value of $\chi^2_{\mathrm{HSIC}}\left(f_j', r_i^{(-i)}\right)$.

$$\chi^2_{\mathrm{HSIC+Fisher}}\left(\mathbf{f}_{(-i)}', r_i^{(-i)}\right) = -2\Sigma_{i\neq j} \log\left\{p_{\mathrm{HSIC}}\left(f_j', r_i^{(-i)}\right)\right\}$$

    d. Find a f_m' whose $\mathbf{f}_{(-m)}'$ and $r_m^{(-m)}$ are independent the most by:

$$f_m' = \arg\max_{i\in U\setminus K} p_{\mathrm{HSIC+Fisher}}\left(\mathbf{f}_{(-i)}', r_i^{(-i)}\right)$$

    where $p_{\mathrm{HSIC+Fisher}}$ is the p-value of $\chi^2_{\mathrm{HSIC+Fisher}}$.
    e. Add the latent-variable subscript m to the top of K. Update $\mathbf{f}' \coloneqq \mathbf{f}_{(-m)}'$ and $U' \coloneqq U'\setminus m$. If $|U'| = 1$, break.
3. Initialize a ($q \times q$) null matrix as $\mathbf{B}$. Reset $U' \coloneqq U$. Repeat

a. Define η_i where $k(i) < k(j)$ for all $j \in U'$ $(i \neq j)$.

b. Gather $\boldsymbol{\eta}_{(h)} = \{\eta_h\}$ where $k(h) < k(i)$ for all $h \in U$ $(i \neq h)$. Should $\boldsymbol{\eta}_{(h)}$ be empty, skip to Step 3d.

c. Run a regularized structural equation modelling of η_i on $\boldsymbol{\eta}_{(h)}$. Gather a vector of structural parameters on $\boldsymbol{\eta}_{(h)}$ to the row corresponding to η_i in **B**.

d. Update $U' \coloneqq U' \backslash i$. If $|U'| = 0$, break and return **B**.

4. Run a regularized structural equation modelling of $\boldsymbol{\eta}$, where the directional relations between latent variables are specified in accordance with **B**. The respective measurement model copies the one that generated $\boldsymbol{\Lambda}$ and $\boldsymbol{\Phi}$. Retrieve a $(q \times q)$ structural parameter matrix as $\widetilde{\mathbf{B}}$.

Output: A $(q \times q)$ structural parameter matrix as $\widetilde{\mathbf{B}}$

Appendix 12: R Documentation of LiNGAM Algorithms

LiNGAM-LV algorithms introduced in this thesis were coded with the R6 class. One can conceive LiNGAM-LV algorithms as some objects in the form of R6 classes. These objects receive inputs (e.g., data and hyperparameters) and produce outputs (e.g., causal orders and discovered path models). We had prepared a total of six R6 classes, each for one LiNGAM algorithm (i.e., ICA-LiNGAM, DirectLiNGAM, ParceLiNGAM, ICA-LiNGAM-LV, DirectLiNGAM-LV & ParceLiNGAM-LV). Let N be the number of observations, p be the number of observed variables, and q be the number of latent variables.

ICALiNGAM

R6 class "`ICALiNGAM`" is to implement ICA-LiNGAM with four editable fields:

`max_k`	A single number that specifies the maximum attempt to search the causal order (Default: 3).
`max_iter`	A single number that specifies the maximum iteration of ICA (Default: 1000).
`type_ica`	A character that assigns a R package between "`fastICA`" and "`ica`" to run ICA (Default: "`fastICA`")
`lasso_engine`	A character that assigns a R package between "`glmnet`" and "`lars`" to run the lasso regression (Default: "`glmnet`").

Method "`fit()`" initiates the algorithm, in which argument "`X`" is a ($N \times p$) data frame. "`ICALiNGAM`" outputs three fields:

`causal_order`	A ($p \times 1$) vector of numbers that sequences the observed variables in a causal order.
`intercept`	A ($p \times 1$) vector of observed variable intercepts.
`adjacency_matrix`	A discovered ($p \times p$) regression coefficient matrix.

DirectLiNGAM

R6 class "`DirectLiNGAM`" is to implement DirectLiNGAM with one editable field:

`lasso_engine`	A character that assigns a R package between "`glmnet`" and "`lars`" to run the lasso regression (Default: "`glmnet`").

Method "`fit()`" initiates the algorithm, in which argument "`X`" is a ($N \times p$) data frame. "`DirectLiNGAM`" outputs three fields:

`causal_order`	A ($p \times 1$) vector of numbers that sequences the observed variables in a causal order.
`intercept`	A ($p \times 1$) vector of observed variable intercepts.
`adjacency_matrix`	A discovered ($p \times p$) regression coefficient matrix.

ParceLiNGAM

R6 class "`ParceLiNGAM`" is to implement ParceLiNGAM with two editable fields:

`alpha`	A single number that specifies the uncorrected threshold value for the Fisher's independent test (Default: 0.001).
`lasso_engine`	A character that assigns a R package between "`glmnet`" and "`lars`" to run the lasso regression (Default: "`glmnet`").

Method "`fit()`" initiates the algorithm, in which argument "`X`" is a ($N \times p$) data frame. "`ParceLiNGAM`" outputs three fields:

`causal_order`	A ($p \times 1$) vector of numbers that sequences the observed variables in a causal order.
`intercept`	A ($p \times 1$) vector of observed variable intercepts.
`adjacency_matrix`	A discovered ($p \times p$) regression coefficient matrix.

ICALiNGAM_LV

R6 class "`ICALiNGAM_LV`" is to implement ICA-LiNGAM-LV with eight editable fields:

`lambda`	A ($p \times q$) factor loading matrix.
`model_pure`	A ($q \times 1$) vector of characters that specifies pure measurement model in which each of the elements is a lavaan model syntax describing the relationship of a latent variable with the corresponding observed variables. The order of elements follows the column of "`lambda`".
`model_other`	A character that specifies lavaan model syntax about configuration of the measurement model that is not yet declared in "model_pure" (e.g., cross-factor loading).
`max_k`	A single number that specifies the number of attempts to search the causal order (Default: 3).
`max_iter_ica`	A single number that specifies the maximum iteration of ICA (Default: 1000).

`type_ica`	A character that assigns a R package between "`fastICA`" or "`ica`" to run ICA (Default: "`fastICA`").
`regsem_penalty`	A character that specifies a penalty type used in a R package of regsem ("`none`", "`lasso`", "`ridge`", "`enet`", "`alasso`", "`diff_lasso`") (Default: "`alasso`").
`regsem_nlambda`	A single number that specifies the number of penalty values to test in "`cvregsem`" function in "`regsem`" package (Default: 20).
`regsem_jump`	A single number that specifies the change in penalty values per iteration in "`cvregsem`" function in "`regsem`" package (Default: .03).

Method "`$fit()`" initiates the algorithm, in which argument "`X`" is a ($N \times p$) data frame that specifies the observed data. "ICALiNGAM_LV" outputs two fields:

`causal_order`	A ($q \times 1$) vector of numbers that sequences the indices of latent variables in a causal order, where the indices correspond to the column of "`lambda`".
`adjacency_matrix`	A discovered ($q \times q$) structural parameter matrix whose row and column follow the column of "`lambda`".

DirectLiNGAM_LV

R6 class "`DirectLiNGAM_LV`" is to implement DirectLiNGAM-LV with nine editable fields:

`lambda`	A ($p \times q$) factor loading matrix.
`phi`	A ($q \times q$) factor covariance matrix.
`model_pure`	A ($q \times 1$) vector of characters that specifies pure measurement model in which each of the elements is a lavaan model syntax describing the relationship of a latent variable with the corresponding observed variables. The order of elements follows the column of "`lambda`".
`model_other`	A character that specifies lavaan model syntax about configuration of the measurement model that is not yet declared in "`model_pure`" (e.g., cross-factor loading).
`fsa`	A vector of characters that specifies the factor scores approximation approaches used in "`psych`" package ("`Thurstone`",

	"tenBerge", "Anderson", "Bartlett", "Harman", "components") (Default: "Thurstone", "Bartlett" & "tenBerge").
fsa_user	A character that specifies one factor score approximation approach to endorse if the causal orders estimated are not consistent (Default: "tenBerge").
regsem_penalty	A character that specifies a penalty type used in "regsem" package ("none", "lasso", "ridge", "enet", "alasso", "diff_lasso") (Default: "alasso").
regsem_nlambda	A single number that specifies the number of penalty values to test in "cvregsem" function in "regsem" package (Default: 20).
regsem_jump	A single number that specifies the change in penalty values per iteration in "cvregsem" function in "regsem" package (Default: 0.03).

Method "fit()" initiates the algorithm, in which argument "X" is a ($N \times p$) data frame that specifies the observed data. "DirectLiNGAM_LV" produces two fields:

causal_order	A ($q \times 1$) vector of numbers that sequences the indices of latent variables in a causal order, where the indices correspond to the column of "lambda".
adjacency_matrix	A discovered ($q \times q$) structural parameter matrix whose row and column follow the column of "lambda".

ParceLiNGAM_LV

R6 class "ParceLiNGAM_LV" is to implement ParceLiNGAM-LV with ten editable fields:

lambda	A ($p \times q$) factor loading matrix.
phi	A ($q \times q$) factor covariance matrix.
model_pure	A ($q \times 1$) vector of characters that specifies pure measurement model in which each of the elements is a lavaan model syntax describing the relationship of a latent variable with the corresponding observed variables. The order of elements follows the column of "lambda".

`model_other`	A character that specifies lavaan model syntax about configuration of the measurement model that is not yet declared in "`model_pure`" (e.g., cross-factor loading).
`fsa`	A vector of characters that specifies the factor scores approximation approaches used in "`psych`" package ("`Thurstone`", "`tenBerge`", "`Anderson`", "`Bartlett`", "`Harman`", "`components`") (Default: "`Thurstone`", "`Bartlett`" & "`tenBerge`").
`fsa_user`	A character that specifies one factor score approximation approach to endorse if the causal orders estimated are not consistent (Default: "`tenBerge`").
`regsem_penalty`	A character that specifies a penalty type used in "`regsem`" package ("`none`", "`lasso`", "`ridge`", "`enet`", "`alasso`", "`diff_lasso`") (Default: "`alasso`").
`regsem_nlambda`	A single number that specifies the number of penalty values to test in "`cvregsem`" function (Default: 20).
`regsem_jump`	A single number that specifies the change in penalty values per iteration in "`cvregsem`" function (Default: 0.03).
`alpha`	A single number that specifies the uncorrected threshold value for the Fisher's independent test (Default: 0.001).

Method "`fit()`" initiates the algorithm, in which argument "`X`" is a ($N \times p$) data frame that specifies the observed data "`ParceLiNGAM_LV`" output two fields:

`causal_order`	A ($q \times 1$) vector of numbers that sequences the indices of latent variables in a causal order, where the indices correspond to the column of "`lambda`".
`adjacency_matrix`	A discovered ($q \times q$) structural parameter matrix whose row and column follow the column of "`lambda`".

Appendix 13: Mean and Standard Deviation of Performance Indicators

A13.1 Mean and Standard Deviation of RMSEA-P on PM1

			ICA-LiNGAM-LV		DirectLiNGAM-LV		ParceLiNGAM-LV	
Missingness	Non-Normality	Sample Size	M	SD	M	SD	M	SD
MCAR	Normal	100	.022	.052	.019	.047	.016	.046
None	Normal	100	.024	.051	.017	.043	.019	.049
MCAR	Mild	100	.018	.044	.021	.048	.017	.047
None	Mild	100	.026	.053	.027	.055	.017	.046
MCAR	Moderate	100	.025	.056	.023	.051	.017	.046
None	Moderate	100	.026	.055	.028	.058	.021	.050
MCAR	Severe	100	.019	.046	.025	.060	.019	.049
None	Severe	100	.020	.047	.023	.049	.017	.042
MCAR	Normal	200	.019	.037	.018	.036	.016	.034
None	Normal	200	.018	.037	.017	.036	.012	.031
MCAR	Mild	200	.019	.038	.020	.039	.013	.032
None	Mild	200	.021	.041	.023	.042	.016	.037
MCAR	Moderate	200	.021	.040	.021	.039	.016	.036
None	Moderate	200	.020	.039	.021	.040	.017	.037
MCAR	Severe	200	.021	.038	.022	.039	.016	.034
None	Severe	200	.017	.036	.019	.037	.015	.033
MCAR	Normal	500	.014	.026	.015	.026	.014	.025
None	Normal	500	.015	.027	.015	.027	.015	.028
MCAR	Mild	500	.015	.028	.016	.028	.014	.027
None	Mild	500	.015	.026	.016	.027	.014	.026
MCAR	Moderate	500	.014	.025	.015	.026	.014	.025
None	Moderate	500	.012	.024	.012	.024	.011	.024
MCAR	Severe	500	.017	.028	.016	.027	.016	.028
None	Severe	500	.015	.028	.016	.028	.015	.028
MCAR	Normal	1000	.010	.018	.010	.018	.010	.018
None	Normal	1000	.010	.019	.010	.019	.010	.019
MCAR	Mild	1000	.011	.020	.011	.020	.011	.019
None	Mild	1000	.013	.021	.013	.021	.012	.021
MCAR	Moderate	1000	.010	.017	.010	.017	.010	.017
None	Moderate	1000	.009	.017	.009	.017	.009	.017
MCAR	Severe	1000	.011	.020	.011	.020	.011	.020
None	Severe	1000	.011	.020	.011	.020	.011	.020

Note: $N = 500$.

A13.2 Mean and Standard Deviation of RMSEA-P on PM2

			ICA-LiNGAM-LV		DirectLiNGAM-LV		ParceLiNGAM-LV	
Missingness	Non-Normality	Sample Size	M	SD	M	SD	M	SD
MCAR	Normal	100	.050	.063	.054	.069	.057	.068
None	Normal	100	.061	.066	.065	.075	.065	.072
MCAR	Mild	100	.052	.066	.050	.068	.059	.074
None	Mild	100	.058	.068	.055	.070	.063	.079
MCAR	Moderate	100	.050	.068	.049	.065	.064	.076
None	Moderate	100	.056	.068	.056	.070	.071	.079
MCAR	Severe	100	.055	.068	.053	.064	.064	.070
None	Severe	100	.058	.074	.048	.067	.068	.078
MCAR	Normal	200	.048	.052	.045	.051	.053	.056
None	Normal	200	.049	.053	.046	.053	.055	.060
MCAR	Mild	200	.043	.057	.040	.047	.060	.062
None	Mild	200	.044	.051	.042	.049	.054	.063
MCAR	Moderate	200	.039	.047	.038	.044	.057	.059
None	Moderate	200	.039	.045	.037	.043	.057	.060
MCAR	Severe	200	.042	.051	.037	.044	.055	.059
None	Severe	200	.043	.049	.038	.045	.055	.058
MCAR	Normal	500	.024	.034	.024	.033	.041	.045
None	Normal	500	.025	.035	.027	.038	.043	.049
MCAR	Mild	500	.025	.034	.025	.033	.046	.045
None	Mild	500	.025	.033	.023	.032	.045	.046
MCAR	Moderate	500	.023	.031	.023	.033	.036	.041
None	Moderate	500	.022	.033	.021	.029	.038	.043
MCAR	Severe	500	.024	.034	.023	.032	.035	.037
None	Severe	500	.024	.036	.022	.032	.035	.039
MCAR	Normal	1000	.011	.023	.012	.023	.015	.027
None	Normal	1000	.008	.020	.009	.021	.013	.027
MCAR	Mild	1000	.009	.019	.008	.018	.035	.040
None	Mild	1000	.009	.022	.009	.022	.032	.043
MCAR	Moderate	1000	.009	.017	.009	.017	.019	.031
None	Moderate	1000	.008	.018	.008	.018	.021	.034
MCAR	Severe	1000	.011	.026	.011	.026	.020	.035
None	Severe	1000	.008	.018	.008	.018	.018	.032

Note: $N = 500$.

A13.3 Mean and Standard Deviation of RMSEA-P on PM3

			ICA-LiNGAM-LV		DirectLiNGAM-LV		ParceLiNGAM-LV	
Missingness	Non-Normality	Sample Size	M	SD	M	SD	M	SD
MCAR	Normal	100	.056	.048	.050	.051	.049	.049
None	Normal	100	.056	.050	.051	.052	.057	.052
MCAR	Mild	100	.057	.049	.042	.047	.051	.050
None	Mild	100	.061	.053	.043	.048	.057	.052
MCAR	Moderate	100	.056	.049	.036	.042	.054	.051
None	Moderate	100	.058	.052	.034	.045	.055	.050
MCAR	Severe	100	.054	.048	.031	.043	.049	.049
None	Severe	100	.057	.051	.027	.038	.051	.052
MCAR	Normal	200	.059	.037	.040	.038	.056	.040
None	Normal	200	.063	.038	.039	.038	.057	.041
MCAR	Mild	200	.056	.039	.028	.034	.055	.041
None	Mild	200	.056	.041	.030	.035	.059	.043
MCAR	Moderate	200	.045	.040	.021	.031	.051	.040
None	Moderate	200	.049	.043	.018	.028	.052	.042
MCAR	Severe	200	.042	.039	.018	.026	.045	.042
None	Severe	200	.048	.045	.017	.027	.050	.042
MCAR	Normal	500	.044	.035	.019	.025	.046	.033
None	Normal	500	.045	.037	.021	.027	.046	.035
MCAR	Mild	500	.029	.032	.012	.017	.050	.035
None	Mild	500	.034	.036	.013	.021	.052	.036
MCAR	Moderate	500	.021	.027	.011	.016	.038	.033
None	Moderate	500	.024	.030	.010	.016	.044	.036
MCAR	Severe	500	.022	.028	.012	.017	.033	.032
None	Severe	500	.020	.028	.012	.016	.032	.033
MCAR	Normal	1000	.021	.029	.013	.020	.027	.029
None	Normal	1000	.018	.026	.010	.017	.026	.029
MCAR	Mild	1000	.011	.017	.008	.012	.036	.034
None	Mild	1000	.010	.015	.007	.011	.037	.037
MCAR	Moderate	1000	.010	.018	.009	.012	.026	.028
None	Moderate	1000	.010	.016	.008	.013	.025	.029
MCAR	Severe	1000	.010	.016	.008	.011	.020	.025
None	Severe	1000	.011	.016	.009	.013	.018	.023

Note: $N = 500$.

A13.4 Mean and Standard Deviation of CFI-P on PM1

			ICA-LiNGAM-LV		DirectLiNGAM-LV		ParceLiNGAM-LV	
Missingness	Non-Normality	Sample Size	M	SD	M	SD	M	SD
MCAR	Normal	100	.994	.018	.995	.017	.996	.016
None	Normal	100	.994	.016	.996	.014	.995	.018
MCAR	Mild	100	.996	.016	.994	.017	.994	.027
None	Mild	100	.994	.016	.993	.019	.995	.017
MCAR	Moderate	100	.992	.034	.994	.020	.995	.019
None	Moderate	100	.994	.017	.993	.019	.995	.016
MCAR	Severe	100	.995	.017	.992	.036	.994	.020
None	Severe	100	.995	.015	.995	.016	.996	.012
MCAR	Normal	200	.997	.009	.997	.008	.997	.008
None	Normal	200	.997	.008	.997	.007	.998	.006
MCAR	Mild	200	.996	.010	.996	.011	.998	.007
None	Mild	200	.996	.010	.996	.010	.997	.008
MCAR	Moderate	200	.996	.010	.996	.009	.997	.008
None	Moderate	200	.997	.009	.996	.009	.997	.009
MCAR	Severe	200	.996	.010	.996	.010	.997	.009
None	Severe	200	.997	.007	.997	.008	.998	.007
MCAR	Normal	500	.998	.004	.998	.004	.998	.004
None	Normal	500	.998	.004	.998	.004	.998	.004
MCAR	Mild	500	.998	.005	.998	.005	.998	.005
None	Mild	500	.998	.004	.998	.004	.998	.004
MCAR	Moderate	500	.998	.004	.998	.004	.998	.004
None	Moderate	500	.999	.004	.999	.004	.999	.004
MCAR	Severe	500	.998	.005	.998	.004	.998	.005
None	Severe	500	.998	.004	.998	.004	.998	.004
MCAR	Normal	1000	.999	.002	.999	.002	.999	.002
None	Normal	1000	.999	.002	.999	.002	.999	.002
MCAR	Mild	1000	.999	.002	.999	.002	.999	.002
None	Mild	1000	.999	.002	.999	.002	.999	.002
MCAR	Moderate	1000	.999	.002	.999	.002	.999	.002
None	Moderate	1000	.999	.002	.999	.002	.999	.002
MCAR	Severe	1000	.999	.003	.999	.003	.999	.003
None	Severe	1000	.999	.002	.999	.002	.999	.002

Note: $N = 500$.

A13.5 Mean and Standard Deviation of CFI-P on PM2

			ICA-LiNGAM-LV		DirectLiNGAM-LV		ParceLiNGAM-LV	
Missingness	Non-Normality	Sample Size	M	SD	M	SD	M	SD
MCAR	Normal	100	.977	.038	.975	.044	.971	.050
None	Normal	100	.977	.036	.972	.045	.973	.047
MCAR	Mild	100	.974	.053	.976	.057	.970	.060
None	Mild	100	.975	.048	.975	.053	.968	.069
MCAR	Moderate	100	.977	.049	.978	.045	.966	.063
None	Moderate	100	.978	.038	.977	.049	.966	.067
MCAR	Severe	100	.979	.036	.978	.040	.974	.037
None	Severe	100	.975	.052	.979	.049	.967	.067
MCAR	Normal	200	.985	.023	.986	.023	.983	.025
None	Normal	200	.986	.024	.987	.020	.983	.026
MCAR	Mild	200	.986	.032	.988	.020	.979	.030
None	Mild	200	.987	.022	.988	.021	.983	.026
MCAR	Moderate	200	.989	.020	.989	.017	.979	.031
None	Moderate	200	.989	.020	.990	.018	.981	.028
MCAR	Severe	200	.987	.025	.989	.022	.980	.033
None	Severe	200	.987	.021	.989	.028	.982	.032
MCAR	Normal	500	.995	.009	.996	.008	.991	.014
None	Normal	500	.996	.008	.995	.009	.990	.014
MCAR	Mild	500	.995	.009	.995	.008	.989	.015
None	Mild	500	.996	.007	.996	.007	.990	.014
MCAR	Moderate	500	.996	.007	.995	.012	.991	.013
None	Moderate	500	.996	.008	.997	.007	.992	.013
MCAR	Severe	500	.995	.009	.996	.008	.992	.013
None	Severe	500	.996	.009	.996	.007	.992	.012
MCAR	Normal	1000	.999	.004	.999	.005	.998	.006
None	Normal	1000	.999	.003	.999	.004	.998	.006
MCAR	Mild	1000	.999	.003	.999	.002	.993	.011
None	Mild	1000	.999	.008	.999	.008	.994	.010
MCAR	Moderate	1000	.999	.002	.999	.002	.996	.008
None	Moderate	1000	.999	.003	.999	.003	.996	.008
MCAR	Severe	1000	.998	.013	.998	.013	.996	.014
None	Severe	1000	.999	.003	.999	.003	.997	.008

Note: $N = 500$.

A13.6 Mean and Standard Deviation of CFI-P on PM3

			ICA-LiNGAM-LV		DirectLiNGAM-LV		ParceLiNGAM-LV	
Missingness	Non-Normality	Sample Size	M	SD	M	SD	M	SD
MCAR	Normal	100	.963	.044	.969	.041	.972	.039
None	Normal	100	.967	.043	.973	.038	.969	.039
MCAR	Mild	100	.960	.049	.974	.040	.970	.041
None	Mild	100	.962	.046	.978	.035	.970	.037
MCAR	Moderate	100	.961	.047	.980	.032	.966	.044
None	Moderate	100	.963	.045	.982	.031	.971	.036
MCAR	Severe	100	.965	.043	.982	.033	.971	.039
None	Severe	100	.963	.047	.986	.026	.971	.046
MCAR	Normal	200	.968	.031	.982	.023	.972	.029
None	Normal	200	.968	.030	.984	.021	.974	.028
MCAR	Mild	200	.968	.033	.989	.019	.973	.029
None	Mild	200	.971	.031	.989	.019	.973	.029
MCAR	Moderate	200	.975	.031	.991	.020	.975	.028
None	Moderate	200	.974	.033	.994	.013	.976	.028
MCAR	Severe	200	.977	.030	.994	.012	.978	.028
None	Severe	200	.974	.032	.994	.012	.977	.027
MCAR	Normal	500	.980	.022	.995	.011	.983	.019
None	Normal	500	.981	.025	.994	.011	.984	.017
MCAR	Mild	500	.988	.018	.997	.006	.980	.020
None	Mild	500	.986	.025	.996	.016	.980	.020
MCAR	Moderate	500	.993	.015	.998	.004	.985	.018
None	Moderate	500	.991	.016	.998	.004	.983	.020
MCAR	Severe	500	.992	.014	.997	.007	.987	.017
None	Severe	500	.993	.020	.998	.004	.989	.016
MCAR	Normal	1000	.992	.016	.997	.009	.992	.012
None	Normal	1000	.994	.012	.998	.006	.993	.010
MCAR	Mild	1000	.998	.006	.999	.002	.987	.016
None	Mild	1000	.998	.005	.999	.002	.987	.022
MCAR	Moderate	1000	.997	.017	.999	.002	.992	.013
None	Moderate	1000	.998	.006	.999	.003	.993	.012
MCAR	Severe	1000	.998	.009	.999	.002	.994	.011
None	Severe	1000	.998	.006	.999	.004	.995	.009

Note: $N = 500$.

A13.7 Mean and Standard Deviation of Accuracy on PM1

			ICA-LiNGAM-LV		DirectLiNGAM-LV		ParceLiNGAM-LV	
Missingness	Non-Normality	Sample Size	M	SD	M	SD	M	SD
MCAR	Normal	100	.819	.234	.755	.246	.711	.252
None	Normal	100	.829	.229	.748	.259	.704	.260
MCAR	Mild	100	.846	.215	.877	.210	.728	.249
None	Mild	100	.862	.209	.888	.195	.735	.244
MCAR	Moderate	100	.880	.194	.910	.176	.748	.238
None	Moderate	100	.908	.174	.919	.175	.781	.242
MCAR	Severe	100	.900	.175	.916	.173	.801	.233
None	Severe	100	.890	.183	.938	.146	.816	.222
MCAR	Normal	200	.921	.172	.917	.181	.835	.237
None	Normal	200	.920	.180	.909	.190	.828	.236
MCAR	Mild	200	.939	.154	.962	.124	.828	.228
None	Mild	200	.957	.128	.971	.113	.840	.228
MCAR	Moderate	200	.965	.111	.978	.091	.876	.208
None	Moderate	200	.959	.120	.970	.112	.860	.212
MCAR	Severe	200	.981	.084	.990	.064	.886	.197
None	Severe	200	.956	.122	.983	.080	.900	.191
MCAR	Normal	500	.979	.093	.974	.108	.967	.124
None	Normal	500	.980	.090	.982	.093	.974	.110
MCAR	Mild	500	.992	.055	.998	.032	.957	.138
None	Mild	500	.991	.056	1.000	.008	.953	.143
MCAR	Moderate	500	.992	.053	1.000	.006	.963	.127
None	Moderate	500	.996	.038	.999	.024	.961	.129
MCAR	Severe	500	.996	.037	.999	.022	.963	.115
None	Severe	500	.997	.030	1.000	.008	.964	.108
MCAR	Normal	1000	.997	.041	.996	.043	.996	.045
None	Normal	1000	.998	.029	.993	.057	.998	.032
MCAR	Mild	1000	1.000	.000	1.000	.000	.993	.055
None	Mild	1000	1.000	.000	1.000	.000	.995	.050
MCAR	Moderate	1000	1.000	.000	1.000	.000	.972	.081
None	Moderate	1000	1.000	.006	1.000	.006	.966	.093
MCAR	Severe	1000	1.000	.006	1.000	.006	.871	.126
None	Severe	1000	.999	.024	1.000	.000	.860	.124

Note: $N = 500$.

A13.8 Mean and Standard Deviation of Accuracy on PM2

			ICA-LiNGAM-LV		DirectLiNGAM-LV		ParceLiNGAM-LV	
Missingness	Non-Normality	Sample Size	M	SD	M	SD	M	SD
MCAR	Normal	100	.732	.164	.642	.168	.610	.168
None	Normal	100	.755	.163	.658	.168	.610	.170
MCAR	Mild	100	.732	.156	.733	.161	.600	.173
None	Mild	100	.735	.163	.746	.162	.617	.168
MCAR	Moderate	100	.756	.156	.764	.154	.636	.165
None	Moderate	100	.756	.156	.777	.148	.642	.169
MCAR	Severe	100	.737	.163	.784	.148	.665	.157
None	Severe	100	.748	.162	.791	.147	.669	.167
MCAR	Normal	200	.809	.160	.776	.181	.695	.182
None	Normal	200	.817	.153	.778	.173	.675	.175
MCAR	Mild	200	.840	.147	.853	.137	.662	.176
None	Mild	200	.836	.153	.854	.146	.656	.181
MCAR	Moderate	200	.855	.144	.885	.120	.725	.167
None	Moderate	200	.873	.131	.889	.116	.729	.174
MCAR	Severe	200	.863	.136	.886	.115	.744	.165
None	Severe	200	.871	.130	.897	.106	.761	.162
MCAR	Normal	500	.910	.132	.924	.119	.815	.167
None	Normal	500	.915	.131	.925	.123	.819	.176
MCAR	Mild	500	.942	.096	.949	.085	.770	.176
None	Mild	500	.951	.081	.955	.080	.776	.172
MCAR	Moderate	500	.960	.066	.962	.061	.836	.161
None	Moderate	500	.956	.071	.960	.062	.841	.153
MCAR	Severe	500	.954	.084	.966	.061	.867	.145
None	Severe	500	.950	.082	.962	.059	.863	.142
MCAR	Normal	1000	.939	.091	.951	.071	.904	.126
None	Normal	1000	.941	.083	.953	.058	.919	.105
MCAR	Mild	1000	.960	.043	.959	.050	.821	.155
None	Mild	1000	.958	.050	.959	.044	.849	.151
MCAR	Moderate	1000	.963	.043	.965	.037	.900	.123
None	Moderate	1000	.960	.040	.959	.042	.903	.109
MCAR	Severe	1000	.960	.043	.961	.040	.910	.106
None	Severe	1000	.960	.042	.961	.039	.908	.108

Note: $N = 500$.

A13.9 Mean and Standard Deviation of Accuracy on PM3

			ICA-LiNGAM-LV		DirectLiNGAM-LV		ParceLiNGAM-LV	
Missingness	Non-Normality	Sample Size	M	SD	M	SD	M	SD
MCAR	Normal	100	.726	.114	.695	.145	.665	.140
None	Normal	100	.727	.123	.701	.139	.673	.139
MCAR	Mild	100	.747	.115	.795	.138	.674	.137
None	Mild	100	.745	.110	.790	.142	.680	.131
MCAR	Moderate	100	.767	.113	.852	.129	.683	.141
None	Moderate	100	.767	.123	.868	.115	.695	.136
MCAR	Severe	100	.769	.119	.885	.109	.718	.136
None	Severe	100	.774	.120	.893	.107	.717	.132
MCAR	Normal	200	.769	.119	.827	.149	.731	.148
None	Normal	200	.772	.120	.837	.147	.728	.144
MCAR	Mild	200	.799	.112	.910	.107	.719	.145
None	Mild	200	.802	.119	.915	.108	.717	.136
MCAR	Moderate	200	.852	.118	.950	.077	.758	.131
None	Moderate	200	.856	.113	.963	.069	.778	.135
MCAR	Severe	200	.872	.111	.963	.066	.810	.128
None	Severe	200	.868	.119	.975	.052	.816	.129
MCAR	Normal	500	.884	.117	.967	.076	.848	.128
None	Normal	500	.881	.119	.963	.083	.859	.131
MCAR	Mild	500	.934	.097	.991	.032	.832	.129
None	Mild	500	.928	.098	.991	.032	.824	.136
MCAR	Moderate	500	.963	.081	.995	.023	.884	.121
None	Moderate	500	.957	.084	.996	.017	.873	.117
MCAR	Severe	500	.961	.083	.997	.017	.919	.100
None	Severe	500	.972	.067	.997	.012	.924	.097
MCAR	Normal	1000	.955	.082	.983	.048	.920	.107
None	Normal	1000	.958	.078	.983	.042	.925	.104
MCAR	Mild	1000	.983	.041	.994	.016	.893	.118
None	Mild	1000	.983	.036	.994	.011	.899	.111
MCAR	Moderate	1000	.990	.028	.995	.009	.943	.083
None	Moderate	1000	.986	.035	.994	.013	.942	.082
MCAR	Severe	1000	.989	.031	.995	.009	.960	.069
None	Severe	1000	.986	.036	.993	.010	.964	.064

Note: $N = 500$.

A13.10 Mean, Standard Deviation and Number of Replications of RMSE on PM1

			ICA-LiNGAM-LV			DirectLiNGAM-LV			ParceLiNGAM-LV		
Missingness	Non-Normality	Sample Size	M	SD	*N*	M	SD	*N*	M	SD	*N*
MCAR	Normal	100	.199	.122	296	.202	.127	235	.200	.116	196
None	Normal	100	.191	.105	303	.190	.117	240	.183	.119	199
MCAR	Mild	100	.204	.133	313	.202	.128	359	.207	.130	207
None	Mild	100	.198	.136	332	.193	.130	364	.191	.112	209
MCAR	Moderate	100	.217	.128	349	.210	.130	385	.207	.132	215
None	Moderate	100	.194	.127	379	.195	.130	400	.193	.122	258
MCAR	Severe	100	.207	.132	363	.203	.127	391	.200	.132	268
None	Severe	100	.189	.131	352	.184	.116	412	.179	.114	281
MCAR	Normal	200	.139	.078	407	.140	.079	407	.139	.078	329
None	Normal	200	.133	.070	411	.132	.070	401	.135	.069	321
MCAR	Mild	200	.146	.081	423	.145	.079	452	.143	.080	310
None	Mild	200	.132	.075	446	.132	.074	465	.130	.074	330
MCAR	Moderate	200	.141	.082	447	.142	.083	466	.142	.081	360
None	Moderate	200	.127	.078	441	.128	.079	459	.127	.080	343
MCAR	Severe	200	.144	.080	474	.142	.078	487	.139	.077	371
None	Severe	200	.138	.080	440	.135	.078	476	.131	.076	387
MCAR	Normal	500	.082	.044	474	.083	.044	472	.084	.044	467
None	Normal	500	.078	.045	475	.079	.045	479	.079	.045	472
MCAR	Mild	500	.086	.045	488	.085	.045	497	.085	.044	454
None	Mild	500	.079	.047	487	.079	.046	498	.078	.046	449
MCAR	Moderate	500	.081	.048	489	.081	.047	499	.080	.047	459
None	Moderate	500	.079	.050	493	.079	.050	497	.078	.048	456
MCAR	Severe	500	.084	.047	495	.084	.047	499	.083	.046	446
None	Severe	500	.085	.046	495	.085	.047	498	.083	.045	442
MCAR	Normal	1000	.057	.031	496	.057	.031	494	.056	.031	495
None	Normal	1000	.055	.028	497	.055	.028	493	.055	.028	498
MCAR	Mild	1000	.058	.033	500	.058	.033	500	.058	.033	492
None	Mild	1000	.057	.033	500	.057	.033	500	.056	.032	495
MCAR	Moderate	1000	.063	.035	500	.063	.035	500	.063	.036	444
None	Moderate	1000	.058	.033	499	.058	.033	499	.057	.031	436
MCAR	Severe	1000	.059	.032	499	.059	.032	499	.060	.034	244
None	Severe	1000	.059	.031	498	.059	.031	500	.057	.031	219

A13.11 Mean, Standard Deviation and Number of Replications of RMSE on PM2

			ICA-LiNGAM-LV			DirectLiNGAM-LV			ParceLiNGAM-LV		
Missingness	Non-Normality	Sample Size	M	SD	*N*	M	SD	*N*	M	SD	*N*
MCAR	Normal	100	.297	.186	44	.227	.114	15	.265	.130	9
None	Normal	100	.272	.106	60	.257	.091	26	.248	.088	13
MCAR	Mild	100	.227	.104	118	.228	.098	107	.216	.111	51
None	Mild	100	.235	.112	121	.251	.122	105	.250	.110	40
MCAR	Moderate	100	.208	.084	268	.205	.085	281	.206	.084	147
None	Moderate	100	.196	.087	276	.197	.084	284	.192	.080	171
MCAR	Severe	100	.181	.069	211	.179	.070	219	.180	.072	175
None	Severe	100	.176	.069	195	.179	.072	209	.182	.071	175
MCAR	Normal	200	.275	.132	33	.286	.131	45	.299	.158	14
None	Normal	200	.263	.130	49	.276	.144	58	.273	.120	18
MCAR	Mild	200	.240	.123	155	.240	.119	163	.233	.096	38
None	Mild	200	.238	.095	143	.242	.113	163	.257	.102	36
MCAR	Moderate	200	.207	.087	300	.207	.088	310	.212	.085	110
None	Moderate	200	.200	.079	300	.201	.079	311	.202	.083	110
MCAR	Severe	200	.181	.071	218	.181	.070	219	.180	.073	94
None	Severe	200	.184	.070	225	.184	.070	225	.184	.068	129
MCAR	Normal	500	.254	.101	48	.263	.116	55	.292	.235	17
None	Normal	500	.248	.111	47	.262	.128	59	.262	.135	17
MCAR	Mild	500	.241	.103	163	.249	.125	183	.242	.091	64
None	Mild	500	.230	.105	174	.234	.104	187	.241	.116	66
MCAR	Moderate	500	.208	.091	323	.208	.091	327	.211	.091	176
None	Moderate	500	.199	.078	299	.199	.079	304	.195	.079	163
MCAR	Severe	500	.177	.066	242	.177	.066	243	.179	.070	171
None	Severe	500	.179	.065	221	.179	.065	220	.180	.067	163
MCAR	Normal	1000	.263	.142	47	.284	.139	72	.315	.181	18
None	Normal	1000	.259	.108	53	.257	.127	79	.267	.139	25
MCAR	Mild	1000	.242	.122	161	.246	.122	179	.248	.114	67
None	Mild	1000	.239	.105	175	.229	.096	192	.210	.086	78
MCAR	Moderate	1000	.204	.083	330	.205	.086	346	.204	.090	207
None	Moderate	1000	.202	.084	299	.202	.085	314	.199	.076	190
MCAR	Severe	1000	.182	.067	226	.182	.067	228	.181	.065	169
None	Severe	1000	.181	.062	225	.180	.061	226	.179	.060	171

A13.12 Mean, Standard Deviation and Number of Replications of RMSE on PM3

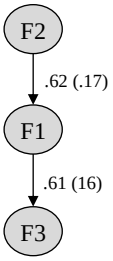
			ICA-LiNGAM-LV			DirectLiNGAM-LV			ParceLiNGAM-LV		
Missingness	Non-Normality	Sample Size	M	SD	*N*	M	SD	*N*	M	SD	*N*
MCAR	Normal	100	.272	.091	12	.309	.112	20	.293	.110	14
None	Normal	100	.288	.104	14	.296	.049	12	.246	.054	6
MCAR	Mild	100	.234	.062	36	.223	.063	121	.211	.068	40
None	Mild	100	.234	.069	35	.212	.068	128	.195	.058	28
MCAR	Moderate	100	.164	.051	192	.154	.050	389	.153	.050	144
None	Moderate	100	.148	.047	195	.140	.046	382	.133	.043	167
MCAR	Severe	100	.106	.033	270	.106	.032	354	.107	.032	250
None	Severe	100	.097	.031	232	.097	.031	324	.098	.030	240
MCAR	Normal	200	.313	.125	19	.304	.080	53	.288	.077	6
None	Normal	200	.289	.067	14	.278	.083	61	.217	.074	8
MCAR	Mild	200	.243	.087	42	.236	.070	204	.241	.073	30
None	Mild	200	.229	.076	64	.222	.066	219	.198	.078	21
MCAR	Moderate	200	.156	.047	291	.155	.048	435	.152	.050	105
None	Moderate	200	.161	.128	274	.154	.105	440	.141	.043	116
MCAR	Severe	200	.106	.034	301	.104	.032	371	.105	.031	192
None	Severe	200	.099	.029	276	.097	.029	359	.096	.029	182
MCAR	Normal	500	.317	.100	16	.294	.097	91	.309	.110	16
None	Normal	500	.293	.090	33	.278	.090	105	.322	.082	13
MCAR	Mild	500	.223	.080	104	.228	.186	269	.212	.065	37
None	Mild	500	.221	.075	104	.220	.074	314	.213	.070	62
MCAR	Moderate	500	.162	.053	364	.154	.050	451	.150	.051	198
None	Moderate	500	.155	.052	355	.148	.049	458	.145	.046	170
MCAR	Severe	500	.107	.035	331	.103	.033	385	.103	.034	275
None	Severe	500	.096	.031	285	.095	.029	355	.095	.030	256
MCAR	Normal	1000	.289	.097	28	.302	.087	133	.310	.091	24
None	Normal	1000	.285	.084	27	.294	.091	132	.307	.096	17
MCAR	Mild	1000	.235	.076	125	.228	.076	292	.227	.074	70
None	Mild	1000	.224	.078	138	.220	.075	345	.211	.073	75
MCAR	Moderate	1000	.163	.055	367	.167	.214	460	.165	.283	255
None	Moderate	1000	.156	.055	376	.148	.052	447	.143	.048	258
MCAR	Severe	1000	.105	.033	328	.104	.033	390	.102	.031	311
None	Severe	1000	.096	.029	285	.095	.029	339	.094	.030	300

Appendix 14: Three Most Frequently Discovered Path Models by Sample Size

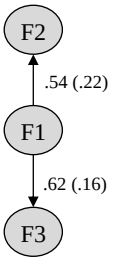
This appendix supplements the main text with the three most frequently discovered path models from LiNGAM-LV algorithms by sample size (i.e., N = 100, 200, 500, 1,000). Given the total number of simulated datasets by sample size was 4,000, the percentage shown below each of the path models represents the likelihood that the corresponding path model was being discovered by the algorithm, and "*n*" denotes the number of simulated datasets that was fed into the algorithm to discover the corresponding path model. The numeric value above the arrow refers to the average parameter estimate (over *n*) and the one within the parenthesis refers to the standard deviation of the parameter.

Three Most Frequently Discovered Models on PM1 at Sample Size of 100

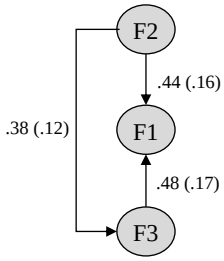
ICA-LiNGAM-LV



1st: 67.13% (*n* = 2687)
RMSEA-P: *M* = .03; *SD* = .06
CFI-P: *M* = .99; *SD* = .02
(Complete match)

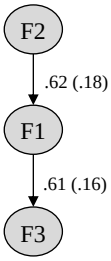


2nd: 17.28% (*n* = 691)
RMSEA-P: *M* = .00; *SD* = .00
CFI-P: *M* = 1.00; *SD* = .00
(F2-F1 pathway in wrong direction)

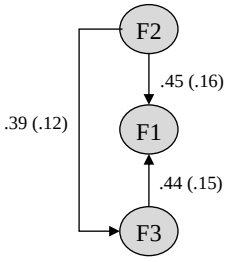


3rd: 5.65% (*n* = 226)
RMSEA-P: *M* = .00; *SD* = .00
CFI-P: *M* = 1.00; *SD* = .00
(F2-F3 pathway extra & F1-F3 pathway in wrong direction)

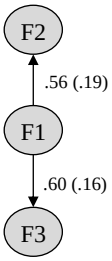
DirectLiNGAM-LV



1st: 69.65% (*n* = 2786)
RMSEA-P: *M* = .03; *SD* = .06
CFI-P: *M* = .99; *SD* = .02
(Complete match)

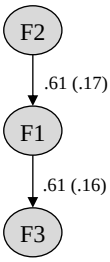


2nd: 9.48% (*n* = 379)
RMSEA-P: *M* = .00; *SD* = .00
CFI-P: *M* = 1.00; *SD* = .00
(F2-F3 pathway extra & F1-F3 pathway in wrong direction)

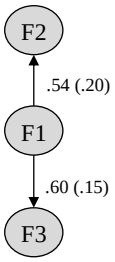


3rd: 7.93% (*n* = 317)
RMSEA-P: *M* = .00; *SD* = .00
CFI-P: *M* = 1.00; *SD* = .00
(F2-F1 pathway in wrong direction)

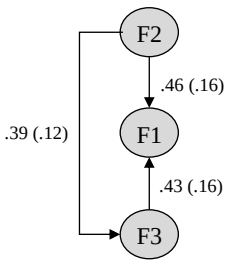
ParceLiNGAM-LV



1st: 45.83% (*n* = 1833)
RMSEA-P: *M* = .03; *SD* = .06
CFI-P: *M* = .99; *SD* = .02
(Complete match)



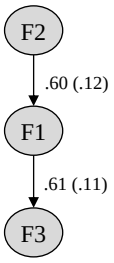
2nd: 17.50% (*n* = 700)
RMSEA-P: *M* = .00; *SD* = .00
CFI-P: *M* = 1.00; *SD* = .00
(F2-F1 pathway in wrong direction)



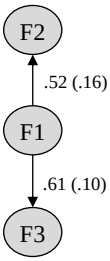
3rd: 17.00% (*n* = 678)
RMSEA-P: *M* = .00; *SD* = .00
CFI-P: *M* = .00; *SD* = .00
(F2-F3 pathway extra & F1-F3 pathway in wrong direction)

Three Most Frequently Discovered Models on PM1 at Sample Size of 200

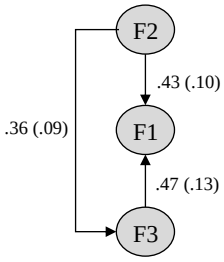
ICA-LiNGAM-LV



1st: 87.23% (*n* = 3489)
RMSEA-P: *M* = .02; *SD* = .04
CFI-P: *M* = 1.00; *SD* = .01
(Complete match)

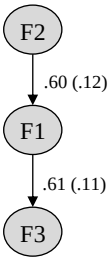


2nd: 8.00% (*n* = 320)
RMSEA-P: *M* = .00; *SD* = .00
CFI-P: *M* = 1.00; *SD* = .00
(F2-F1 pathway in wrong direction)

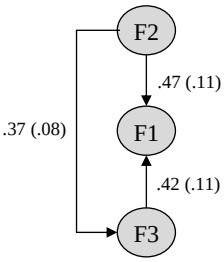


3rd: 2.68% (*n* = 107)
RMSEA-P: *M* = .00; *SD* = .00
CFI-P: *M* = 1.00; *SD* = .00
(F2-F3 pathway extra & F1-F3 pathway in wrong direction)

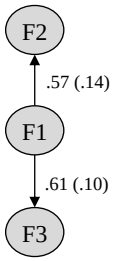
DirectLiNGAM-LV



1st: 90.33% (*n* = 3613)
RMSEA-P: *M* = .02; *SD* = .04
CFI-P: *M* = 1.00; *SD* = .01
(Complete match)

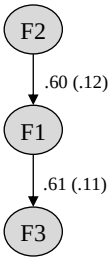


2nd: 4.58% (*n* = 183)
RMSEA-P: *M* = .00; *SD* = .00
CFI-P: *M* = 1.00; *SD* = .00
(F2-F3 pathway extra & F1-F3 pathway in wrong direction)

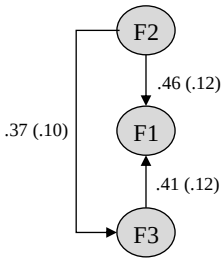


3rd: 2.72% (*n* = 103)
RMSEA-P: *M* = .00; *SD* = .00
CFI-P: *M* = 1.00; *SD* = .00
(F2-F1 pathway in wrong direction)

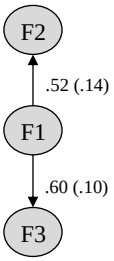
ParceLiNGAM-LV



1st: 68.78% (*n* = 2751)
RMSEA-P: *M* = .02; *SD* = .04
CFI-P: *M* = 1.00; *SD* = .01
(Complete match)



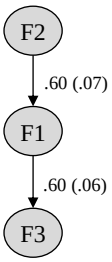
2nd: 16.98% (*n* = 678)
RMSEA-P: *M* = .00; *SD* = .00
CFI-P: *M* = 1.00; *SD* = .00
(F2-F3 pathway extra & F1-F3 pathway in wrong direction)



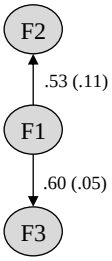
3rd: 9.38% (*n* = 375)
RMSEA-P: *M* = .00; *SD* = .00
CFI-P: *M* = 1.00; *SD* = .00
(F2-F1 pathway in wrong direction)

Three Most Frequently Discovered Models on PM1 at Sample Size of 500

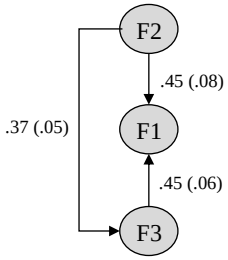
ICA-LiNGAM-LV



1st: 97.40% (*n* = 3896)
RMSEA-P: *M* = .02; *SD* = .04
CFI-P: *M* = 1.00; *SD* = .01
(Complete match)

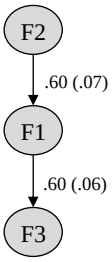


2nd: 1.88% (*n* = 75)
RMSEA-P: *M* = .00; *SD* = .00
CFI-P: *M* = 1.00; *SD* = .00
(F2-F1 pathway in wrong direction)

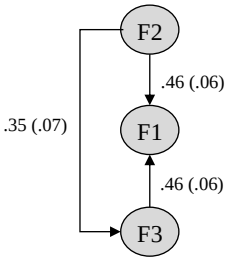


3rd: 0.45% (*n* = 18)
RMSEA-P: *M* = .00; *SD* = .00
CFI-P: *M* = 1.00; *SD* = .00
(F2-F3 pathway extra & F1-F3 pathway in wrong direction)

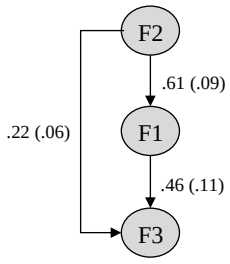
DirectLiNGAM-LV



1st: 98.48% (*n* = 3939)
RMSEA-P: *M* = .02; *SD* = .03
CFI-P: *M* = 1.00; *SD* = .00
(Complete match)

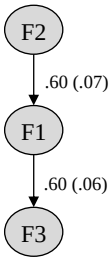


2nd: 0.78% (*n* = 31)
RMSEA-P: *M* = .00; *SD* = .00
CFI-P: *M* = 1.00; *SD* = .00
(F2-F3 pathway extra & F1-F3 pathway in wrong direction)

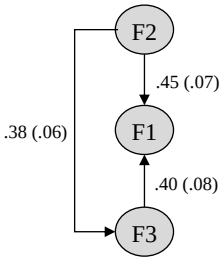


3rd: 0.28% (*n* = 11)
RMSEA-P: *M* = .00; *SD* = .00
CFI-P: *M* = 1.00; *SD* = .00
(F2-F3 pathway extra)

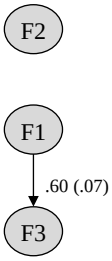
ParceLiNGAM-LV



1st: 91.13% (*n* = 3645)
RMSEA-P: *M* = .02; *SD* = .03
CFI-P: *M* = 1.00; *SD* = .01
(Complete match)



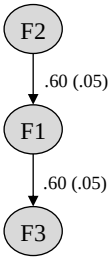
2nd: 5.50% (*n* = 220)
RMSEA-P: *M* = .00; *SD* = .00
CFI-P: *M* = 1.00; *SD* = .00
(F2-F3 pathway extra & F1-F3 pathway in wrong direction)



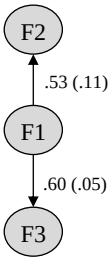
3rd: 1.88% (*n* = 75)
RMSEA-P: *M* = .01; *SD* = .02
CFI-P: *M* = 1.00; *SD* = .00
(F2-F1 pathway missing)

Three Most Frequently Discovered Models on PM1 at Sample Size of 1,000

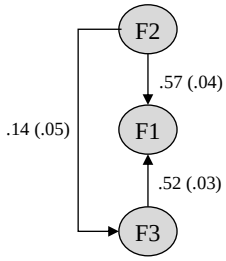
ICA-LiNGAM-LV



1st: 99.73% (*n* = 3989)
RMSEA-P: *M* = .01; *SD* = .02
CFI-P: *M* = 1.00; *SD* = .00
(Complete match)

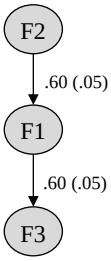


2nd: 0.18% (*n* = 7)
RMSEA-P: *M* = .00; *SD* = .00
CFI-P: *M* = 1.00; *SD* = .00
(F2-F1 pathway in wrong direction)

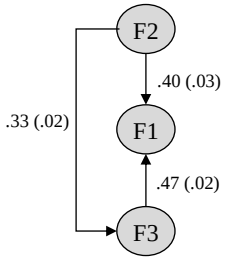


3rd: 0.08% (*n* = 3)
RMSEA-P: *M* = .00; *SD* = .00
CFI-P: *M* = 1.00; *SD* = .00
(F2-F3 pathway extra & F1-F3 pathway in wrong direction)

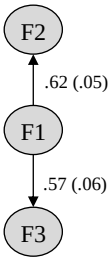
DirectLiNGAM-LV



1st: 99.63% (*n* = 3985)
RMSEA-P: *M* = .01; *SD* = .02
CFI-P: *M* = 1.00; *SD* = .00
(Complete match)

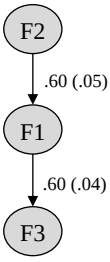


2nd: 0.13% (*n* = 5)
RMSEA-P: *M* = .00; *SD* = .00
CFI-P: *M* = 1.00; *SD* = .00
(F2-F3 pathway extra & F1-F3 pathway in wrong direction)

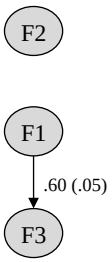


3rd: 0.10% (*n* = 4)
RMSEA-P: *M* = .00; *SD* = .00
CFI-P: *M* = 1.00; *SD* = .00
(F2-F1 pathway in wrong direction)

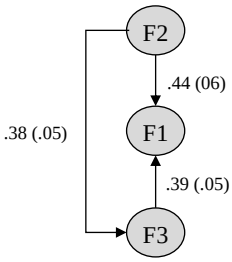
ParceLiNGAM-LV



1st: 83.08% (*n* = 3323)
RMSEA-P: *M* = .01; *SD* = .02
CFI-P: *M* = 1.00; *SD* = .00
(Complete match)



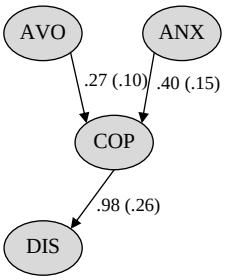
2nd: 16.23% (*n* = 649)
RMSEA-P: *M* = .01; *SD* = .02
CFI-P: *M* = 1.00; *SD* = .00
(F2-F1 pathway missing)



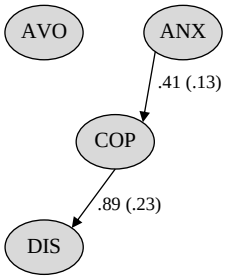
3rd: 0.55% (*n* = 22)
RMSEA-P: *M* = .00; *SD* = .00
CFI-P: *M* = 1.00; *SD* = .00
(F2-F3 pathway extra & F1-F3 pathway in wrong direction)

Three Most Frequently Discovered Models on PM2 at Sample Size of 100

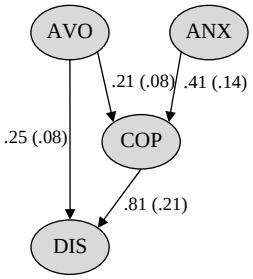
ICA-LiNGAM-LV



1st:16.00% (*n* = 640)
RMSEA-P: *M* = .05; *SD* = .05
CFI-P: *M* = .98; *SD* = .04
(AVO-DIS pathway missing)

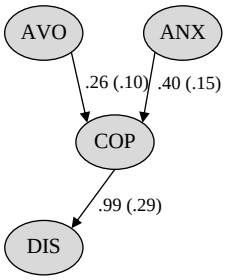


2nd: 9.90% (*n* = 396)
RMSEA-P: *M* = .08; *SD* = .05
CFI-P: *M* = .95; *SD* = .05
(AVO-DIS pathway missing &
AVO-COP pathway missing)

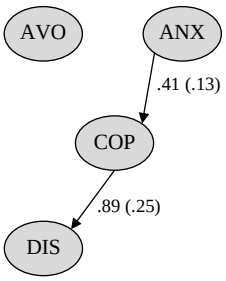


3rd: 9.53% (*n* = 381)
RMSEA-P: *M* = .02; *SD* = .04
CFI-P: *M* = 1.00; *SD* = .01
(Complete match)

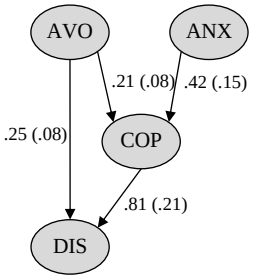
DirectLiNGAM-LV



1st: 15.05% (*n* = 602)
RMSEA-P: *M* = .05; *SD* = .06
CFI-P: *M* = .98; *SD* = .05
(AVO-DIS pathway missing)

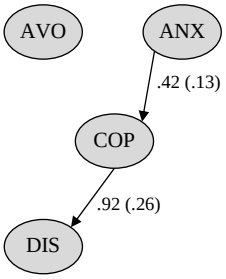


2nd: 10.58% (*n* = 423)
RMSEA-P: *M* = .08; *SD* = .06
CFI-P: *M* = .95; *SD* = .03
(AVO-DIS pathway missing &
AVO-COP pathway missing)

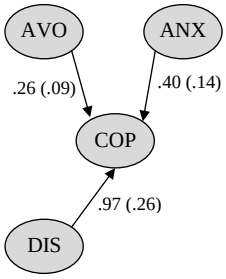


3rd: 10.23% (*n* = 409)
RMSEA-P: *M* = .02; *SD* = .04
CFI-P: *M* = 1.00; *SD* = .01
(Complete match)

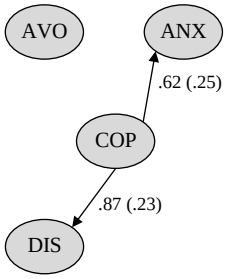
ParceLiNGAM-LV



1st: 7.43% (*n* = 297)
RMSEA-P: *M* = .01; *SD* = .06
CFI-P: *M* = .94; *SD* = .06
(AVO-DIS pathway missing &
AVO-COP pathway missing)



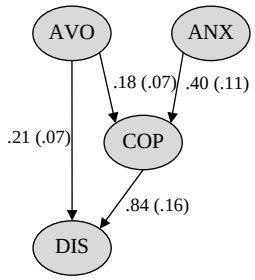
2nd: 5.25% (*n* = 210)
RMSEA-P: *M* = .05; *SD* = .05
CFI-P: *M* = .98; *SD* = .02
(AVO-DIS pathway missing)



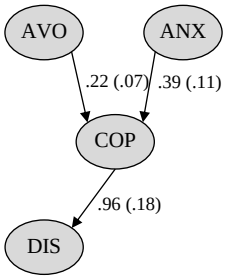
3rd: 4.70% (*n* = 188)
RMSEA-P: *M* = .08; *SD* = .07
CFI-P: *M* = .96; *SD* = .05
(AVO-DIS pathway missing,
AVO-COP pathway missing,
ANX-COP pathway in wrong
direction)

Three Most Frequently Discovered Models on PM2 at Sample Size of 200

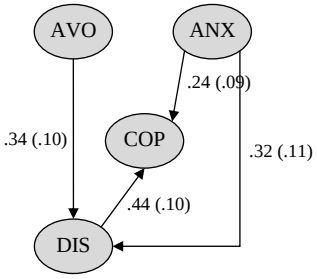
ICA-LiNGAM-LV



1st: 30.25% (*n* = 1210)
RMSEA-P: *M* = .02; *SD* = .03
CFI-P: *M* = 1.00; *SD* = .01
(Complete match)

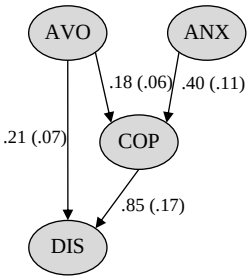


2nd: 23.45% (*n* = 938)
RMSEA-P: *M* = .06; *SD* = .04
CFI-P: *M* = .98; *SD* = .02
(AVO-DIS pathway missing)

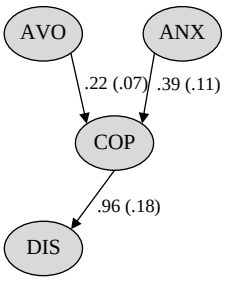


3rd: 8.35% (*n* = 334)
RMSEA-P: *M* = .02; *SD* = .04
CFI-P: *M* = 1.00; *SD* = .01
(AVO-COP pathway missing, COP-DIS pathway in wrong direction & ANX-DIS pathway extra)

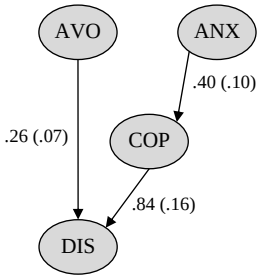
DirectLiNGAM-LV



1st: 31.98% (*n* = 1279)
RMSEA-P: *M* = .02; *SD* = .04
CFI-P: *M* = 1.00; *SD* = .01
(Complete match)

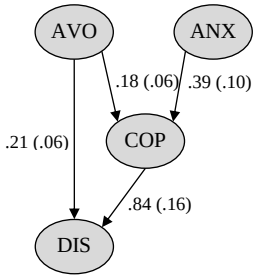


2nd: 24.18% (*n* = 967)
RMSEA-P: *M* = .06; *SD* = .04
CFI-P: *M* = .98; *SD* = .02
(AVO-DIS pathway missing)

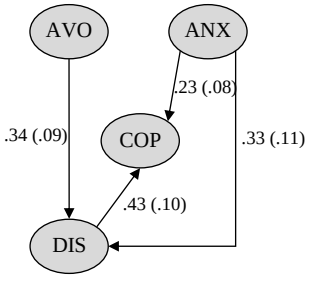


3rd: 7.25% (*n* = 290)
RMSEA-P: *M* = .05; *SD* = .04
CFI-P: *M* = .99; *SD* = .02
(AVO-COP pathway missing)

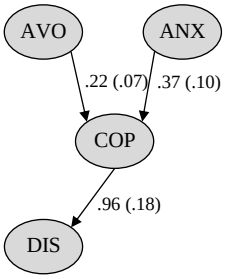
ParceLiNGAM-LV



1st: 11.00% (*n* = 440)
RMSEA-P: *M* = .02; *SD* = .03
CFI-P: *M* = 1.00; *SD* = .01
(Complete match)



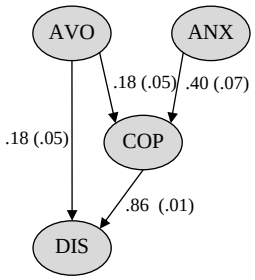
2nd: 8.48% (*n* = 339)
RMSEA-P: *M* = .02; *SD* = .03
CFI-P: *M* = 1.00; *SD* = .00
(AVO-COP pathway missing, COP-DIS pathway in wrong direction & ANX-DIS pathway extra)



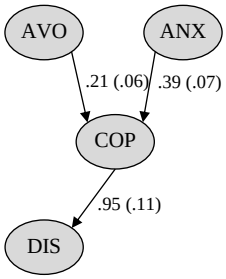
3rd: 8.25% (*n* = 330)
RMSEA-P: *M* = .05; *SD* = .04
CFI-P: *M* = .98; *SD* = .02
(AVO-DIS pathway missing)

Three Most Frequently Discovered Models on PM2 at Sample Size of 500

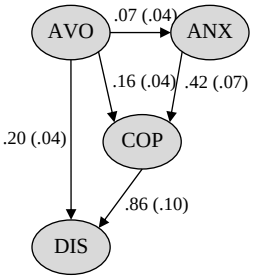
ICA-LiNGAM-LV



1st: 59.88% (*n* = 2395)
RMSEA-P: *M* = .01; *SD* = .03
CFI-P: *M* = 1.00; *SD* = .00
(Complete match)

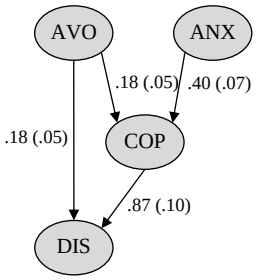


2nd: 13.88% (*n* = 555)
RMSEA-P: *M* = .07; *SD* = .02
CFI-P: *M* = .98; *SD* = .01
(AVO-DIS pathway missing)

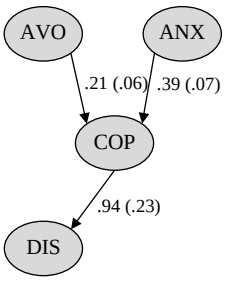


3rd: 13.65% (*n* = 546)
RMSEA-P: *M* = .01; *SD* = .02
CFI-P: *M* = 1.00; *SD* = .00
(AVO-ANX pathway extra)

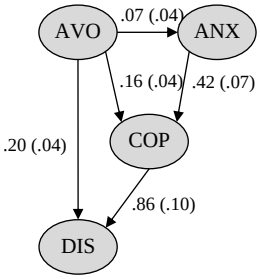
DirectLiNGAM-LV



1st: 61.93% (*n* = 2477)
RMSEA-P: *M* = .01; *SD* = .03
CFI-P: *M* = 1.00; *SD* = .00
(Complete match)

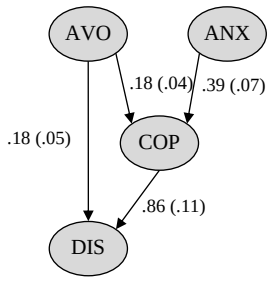


2nd: 14.15% (*n* = 566)
RMSEA-P: *M* = .07; *SD* = .02
CFI-P: *M* = .98; *SD* = .01
(AVO-DIS pathway missing)

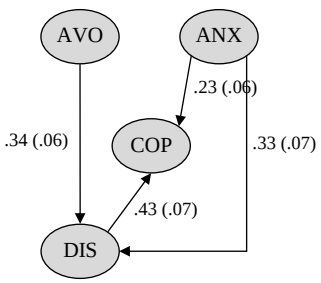


3rd: 7.83% (*n* = 313)
RMSEA-P: *M* = .01; *SD* = .02
CFI-P: *M* = 1.00; *SD* = .00
(AVO-ANX pathway extra)

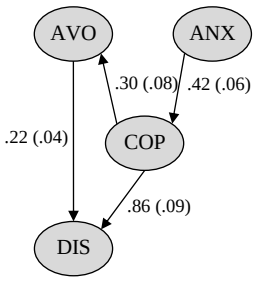
ParceLiNGAM-LV



1st: 31.85% (*n* = 1274)
RMSEA-P: *M* = .01; *SD* = .02
CFI-P: *M* = 1.00; *SD* = .00
(Complete match)



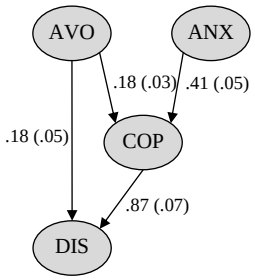
2nd: 10.08% (*n* = 403)
RMSEA-P: *M* = .02; *SD* = .03
CFI-P: *M* = 1.00; *SD* = .00
(ANX-DIS pathway extra, AVO-COP pathway missing, and COP-DIS pathway in wrong direction)



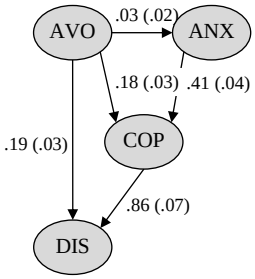
3rd: 9.88% (*n* = 395)
RMSEA-P: *M* = .07; *SD* = .04
CFI-P: *M* = .98; *SD* = .02
(AVO-COP pathway in wrong direction)

Three Most Frequently Discovered Models on PM2 at Sample Size of 1,000

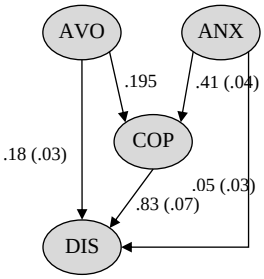
ICA-LiNGAM-LV



1st: 44.08% (*n* = 1763)
RMSEA-P: *M* = .01; *SD* = .02
CFI-P: *M* = 1.00; *SD* = .00
(Complete match)

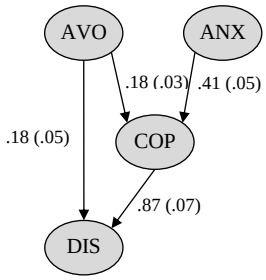


2nd: 35.55% (*n* = 1422)
RMSEA-P: *M* = .01; *SD* = .02
CFI-P: *M* = 1.00; *SD* = .00
(AVO-ANX pathway extra)

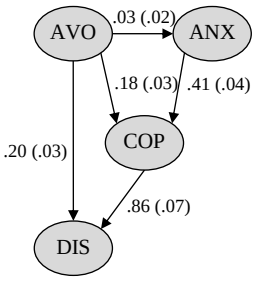


3rd: 11.95% (*n* = 478)
RMSEA-P: *M* = .05; *SD* = .00
CFI-P: *M* = 1.00; *SD* = .00
(ANX-DIS pathway extra)

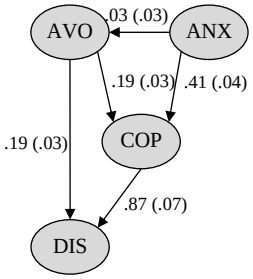
DirectLiNGAM-LV



1st: 44.73% (*n* = 1789)
RMSEA-P: *M* = .01; *SD* = .02
CFI-P: *M* = 1.00; *SD* = .00
(Complete match)

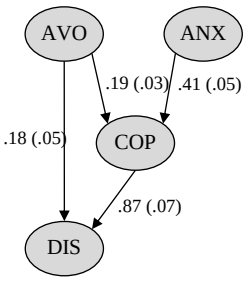


2nd: 20.18% (*n* = 807)
RMSEA-P: *M* = .01; *SD* = .02
CFI-P: *M* = 1.00; *SD* = .00
(AVO-ANX pathway extra)

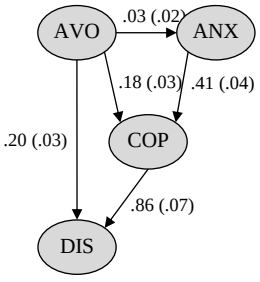


3rd: 15.95% (*n* = 638)
RMSEA-P: *M* = .01; *SD* = .02
CFI-P: *M* = 1.00; *SD* = .00
(ANX-AVO pathway extra)

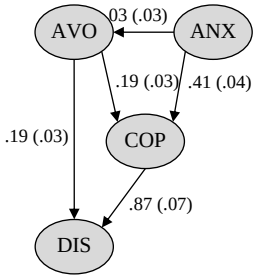
ParceLiNGAM-LV



1st: 31.18% (*n* = 1247)
RMSEA-P: *M* = .01; *SD* = .02
CFI-P: *M* = 1.00; *SD* = .00
(Complete match)



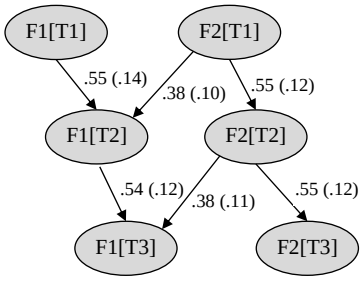
2nd: 16.75% (*n* = 670)
RMSEA-P: *M* = .01; *SD* = .02
CFI-P: *M* = 1.00; *SD* = .00
(AVO-ANX pathway extra)



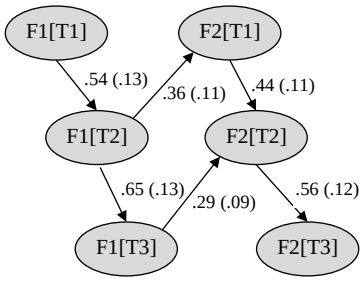
3rd: 10.38% (*n* = 415)
RMSEA-P: *M* = .01; *SD* = .02
CFI-P: *M* = 1.00; *SD* = .00
(AVO-DIS pathway missing)

Three Most Frequently Discovered Models on PM3 at Sample Size of 100

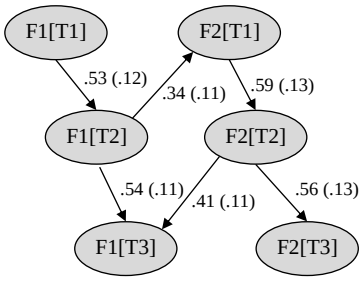
ICA-LiNGAM-LV



1st: 4.08% (n = 163)
RMSEA-P: M = .01; SD = .03
CFI-P: M = .99; SD = .01
(Complete match)

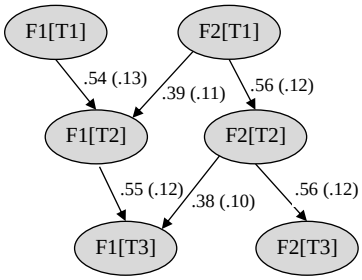


2nd: 3.10% (n = 124)
RMSEA-P: M = .08; SD = .05
CFI-P: M =.94; SD = .06
(F2[T1]-F1[T2] pathway and F2[T2]-F1[T3] pathway in wrong direction)

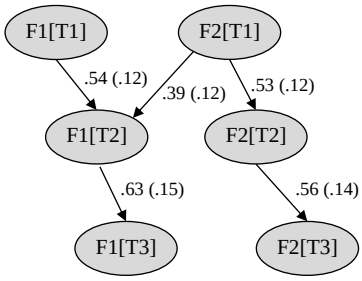


3rd: 2.80% (n = 112)
RMSEA-P: M = .05; SD = .04
CFI-P: M = .97; SD = .03
(F2[T1]-F1[T2] pathway in wrong direction)

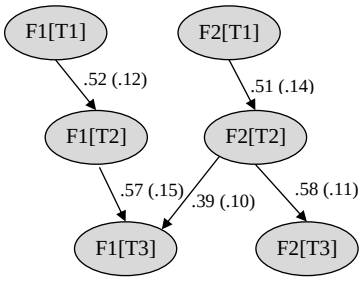
DirectLiNGAM-LV



1st: 15.18% (n = 607)
RMSEA-P: M = .01; SD = .03
CFI-P: M = .99; SD = .01
(Complete match)

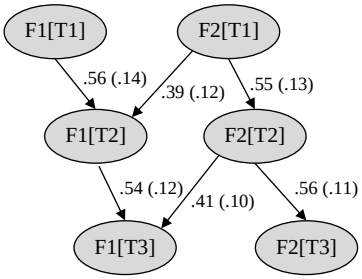


2nd: 3.43% (n = 137)
RMSEA-P: M = .04; SD = .04
CFI-P: M = .98; SD = .03
(F2[T2]-F1[T3] pathway missing)

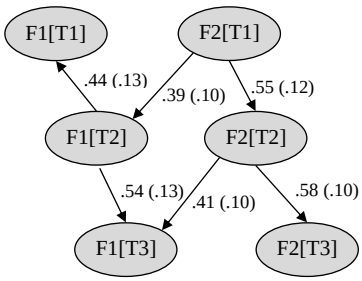


3rd: 2.53% (n = 101)
RMSEA-P: M = .04; SD = .04
CFI-P: M = .97; SD = .03
(F2[T1]-F1[T2] pathway missing)

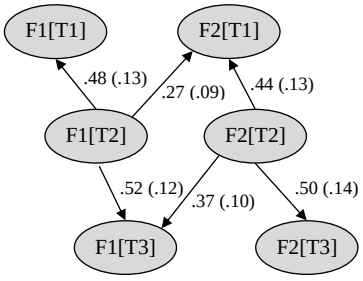
ParceLiNGAM-LV



1st: 2.60% (n = 104)
RMSEA-P: M = .01; SD = .03
CFI-P: M = .99; SD = .01
(Complete match)



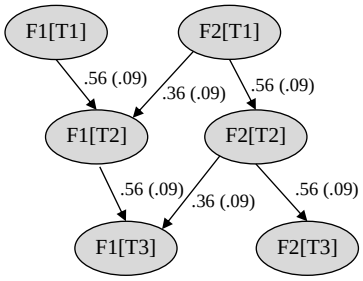
2nd: 1.73% (n = 69)
RMSEA-P: M = .06; SD = .05
CFI-P: M = .97; SD = .04
(F2[T1]-F1[T2] pathway in wrong direction)



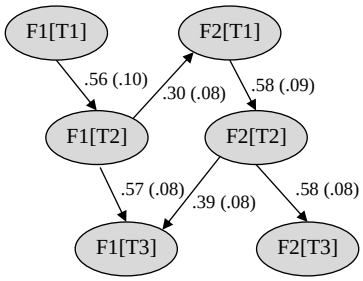
3rd: 1.15% (n = 46)
RMSEA-P: M = .06; SD = .07
CFI-P: M = .98; SD = .02
(F1[T1]-F1[T2], F2[T1]-F2[T2] and F1[T2]-F2[T1] pathways in wrong direction)

Three Most Frequently Discovered Models on PM3 at Sample Size of 200

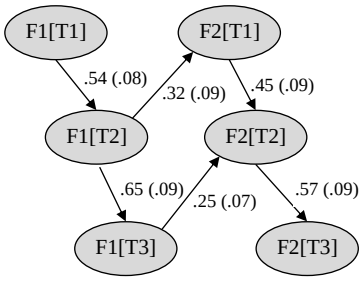
ICA-LiNGAM-LV



1st: 16.20% (*n* = 648)
RMSEA-P: *M* = .01; *SD* = .02
CFI-P: *M* = 1.00; *SD* = .01
(Complete match)

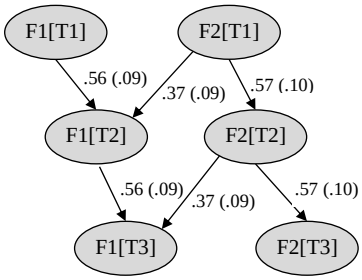


2nd: 7.68% (*n* = 307)
RMSEA-P: *M* = .05; *SD* = .03
CFI-P: *M* =.97; *SD* = .02
(F2[T1]-F1[T2] pathway in wrong direction)

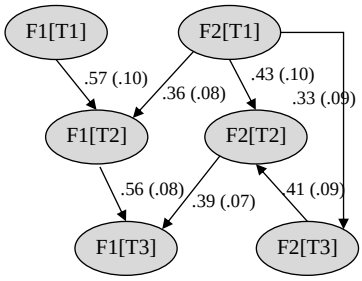


3rd: 4.23% (*n* = 169)
RMSEA-P: *M* = .05; *SD* = .04
CFI-P: *M* = .97; *SD* = .03
(F2[T1]-F1[T2] & F2[T2]-F1[T3] pathways in wrong direction)

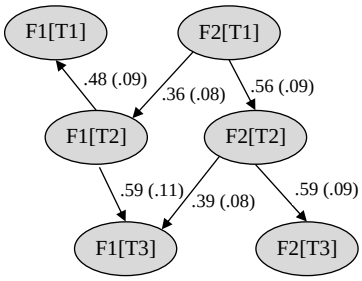
DirectLiNGAM-LV



1st: 47.30% (*n* = 1892)
RMSEA-P: *M* = .01; *SD* = .02
CFI-P: *M* = 1.00; *SD* = .01
(Complete match)

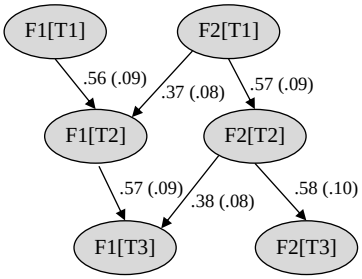


2nd: 2.63% (*n* = 105)
RMSEA-P: *M* = .01; *SD* = .02
CFI-P: *M* = 1.00; *SD* = .01
(F2[T2]-F2[T3] pathway in wrong direction & F2[T1]-F2[T3] pathway extra)

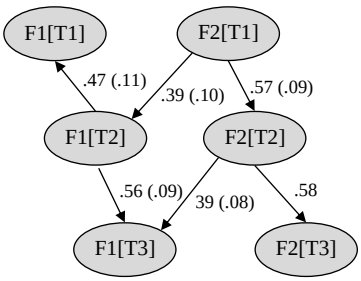


3rd: 1.85% (*n* = 74)
RMSEA-P: *M* = .05; *SD* = .04
CFI-P: *M* = .98; *SD* = .02
(F1[T1]-F1[T2] pathway in wrong direction)

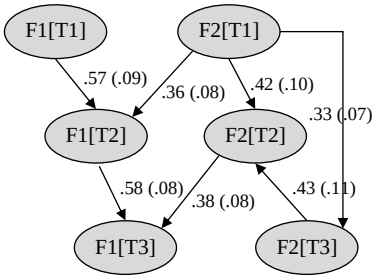
ParceLiNGAM-LV



1st: 9.08% (*n* = 363)
RMSEA-P: *M* = .01; *SD* = .02
CFI-P: *M* = 1.00; *SD* = .01
(Complete match)



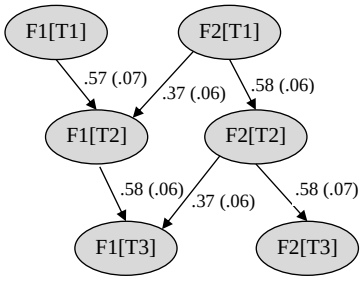
2nd: 3.70% (*n* = 148)
RMSEA-P: *M* = .06; *SD* = .04
CFI-P: *M* = .97; *SD* = .02
(F1[T1]-F1[T2] pathway in wrong direction)



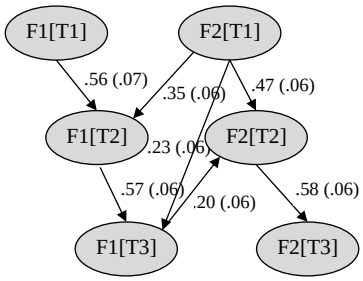
3rd: 1.15% (*n* = 76)
RMSEA-P: *M* = .01; *SD* = .01
CFI-P: *M* = 1.00; *SD* = .01
(F2[T2]-F2[T3] pathway in wrong direction & F2[T2]-F2[T3] pathway extra)

Three Most Frequently Discovered Models on PM3 at Sample Size of 500

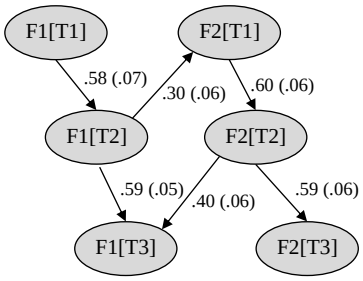
ICA-LiNGAM-LV



1st: 60.35% (*n* = 2414)
RMSEA-P: *M* = .01; *SD* = .02
CFI-P: *M* = 1.00; *SD* = .01
(Complete match)

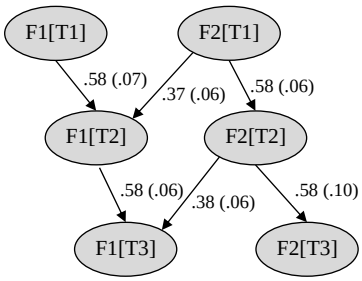


2nd: 5.05% (*n* = 202)
RMSEA-P: *M* = .05; *SD* = .02
CFI-P: *M* =.98; *SD* = .01
(F2[T1]-F1[T3] pathway extra & F2[T2]-F1[T3] pathway in wrong direction)

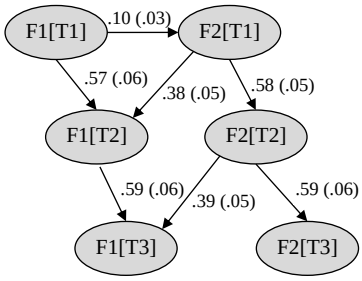


3rd: 4.50% (*n* = 180)
RMSEA-P: *M* = .06; *SD* = .02
CFI-P: *M* = .97; *SD* = .01
(F2[T1]-F1[T2] pathway in wrong direction)

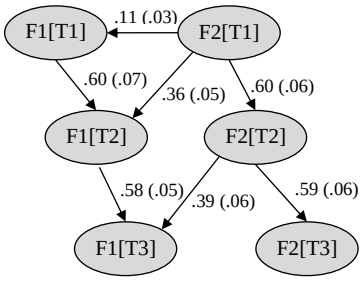
DirectLiNGAM-LV



1st: 86.55% (*n* = 3462)
RMSEA-P: *M* = .01; *SD* = .02
CFI-P: *M* = 1.00; *SD* = .01
(Complete match)

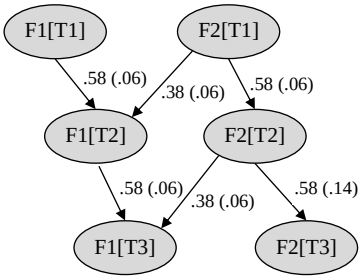


2nd: 2.35% (*n* = 94)
RMSEA-P: *M* = .01; *SD* = .02
CFI-P: *M* = 1.00; *SD* = .00
(F1[T1]-F2[T1] pathway extra)

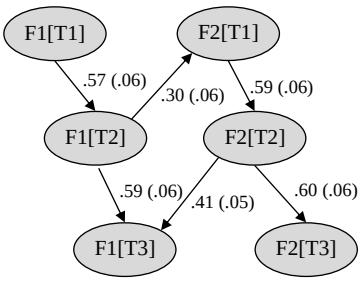


3rd: 1.05% (*n* = 42)
RMSEA-P: *M* = .01; *SD* = .02
CFI-P: *M* = .98; *SD* = .00
(F2[T1]-F1[T1] pathway extra)

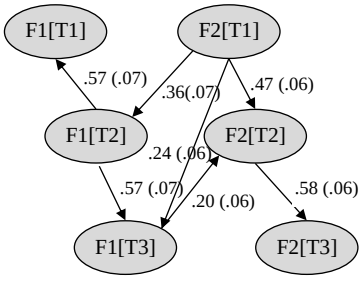
ParceLiNGAM-LV



1st: 35.33% (*n* = 1413)
RMSEA-P: *M* = .01; *SD* = .02
CFI-P: *M* = 1.00; *SD* = .01
(Complete match)



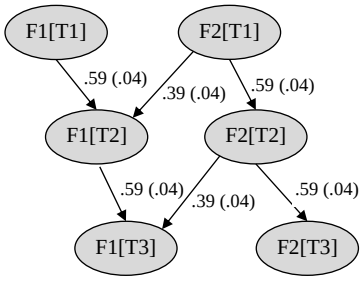
2nd: 4.70% (*n* = 188)
RMSEA-P: *M* = .06; *SD* = .02
CFI-P: *M* = .97; *SD* = .02
(F2[T1]-F1[T2] pathway in wrong direction)



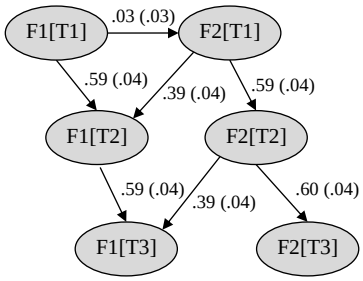
3rd: 3.70% (*n* = 148)
RMSEA-P: *M* = .05; *SD* = .02
CFI-P: *M* = .98; *SD* = .01
(F2[T2]-F1[T3] pathway in wrong direction & F2[T1]-F1[T3] pathway extra)

Three Most Frequently Discovered Models on PM3 at Sample Size of 1,000

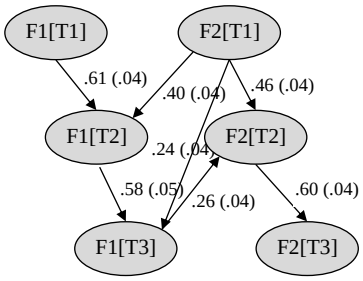
ICA-LiNGAM-LV



1st: 57.70% (*n* = 2308)
RMSEA-P: *M* = .01; *SD* = .01
CFI-P: *M* = 1.00; *SD* = .00
(Complete match)

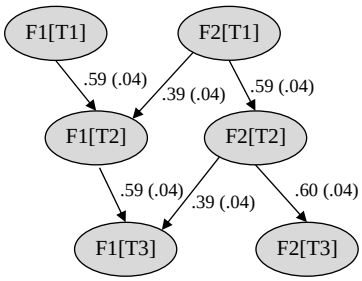


2nd: 33.18% (*n* = 1327)
RMSEA-P: *M* = .01; *SD* = .01
CFI-P: *M* = 1.00; *SD* = .00
(F1[T1]-F2[T1] pathway extra)

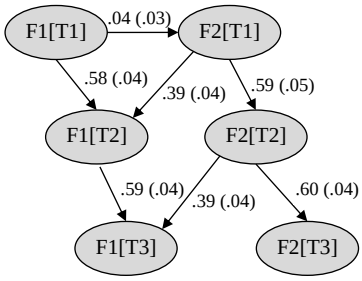


3rd: 1.48% (*n* = 59)
RMSEA-P: *M* = .06; *SD* = .01
CFI-P: *M* = .98; *SD* = .01
(F2[T2]-F1[T3] pathway in wrong direction & F2[T1]-F1[T3] pathway extra)

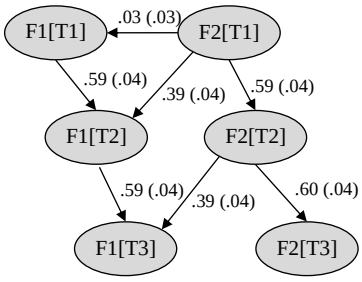
DirectLiNGAM-LV



1st: 71.93% (*n* = 2877)
RMSEA-P: *M* = .01; *SD* = .01
CFI-P: *M* = 1.00; *SD* = .00
(Complete match)

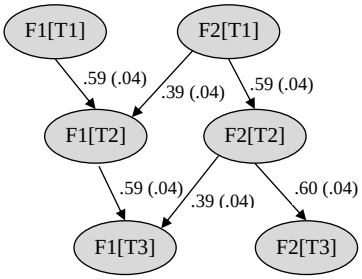


2nd: 13.60% (*n* = 544)
RMSEA-P: *M* = .01; *SD* = .01
CFI-P: *M* = 1.00; *SD* = .00
(F1[T1]-F2[T1] pathway extra)

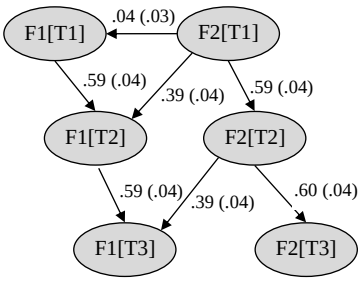


3rd: 10.55% (*n* = 422)
RMSEA-P: *M* = .01; *SD* = .01
CFI-P: *M* = 1.00; *SD* = .00
(F2[T1]-F1[T1] pathway extra)

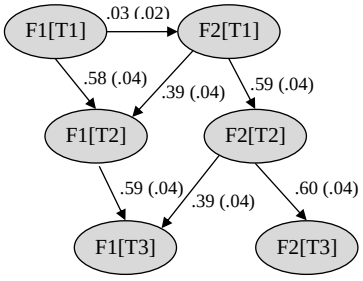
ParceLiNGAM-LV



1st: 50.15% (*n* = 2006)
RMSEA-P: *M* = .01; *SD* = .01
CFI-P: *M* = 1.00; *SD* = .00
(Complete match)



2nd: 6.43% (*n* = 257)
RMSEA-P: *M* = .01; *SD* = .01
CFI-P: *M* = 1.00; *SD* = .00
(F2[T1]-F1[T1] pathway extra)



3rd: 6.23% (*n* = 249)
RMSEA-P: *M* = .01; *SD* = .01
CFI-P: *M* = 1.00; *SD* = .00
(F1[T1]-F2[T1] pathway extra)

Appendix 15: LiNGAM Algorithms with R in Action

This appendix is to demonstrate how to run LiNGAM algorithms on R. R is an open-source software that enables one to conduct statistical analyses. R can be installed in a variety of operation systems, including Windows, Macintosh OS, and Linux. For more information about R, please visit http:///cran.r-project.org/. This appendix assumes readers to have some basic understanding as to how R works, including data reading, R library and variable creation. Those who are interested in learning more about running latent variable models using R are recommended to see Finch and French (2015).

A15.1 Preparation

R codes to run LiNGAM algorithms can be found in Appendix 16. Copy the R code from Appendix 16 to an empty R file on the computer and rename the R file to "LiNGAM_Algorithm_v01.R". Initialize a R session and set the working directory to the location of the file. Then we load it into R with the command below:

```
source("LiNGAM_Algorithm_v01.R")
```

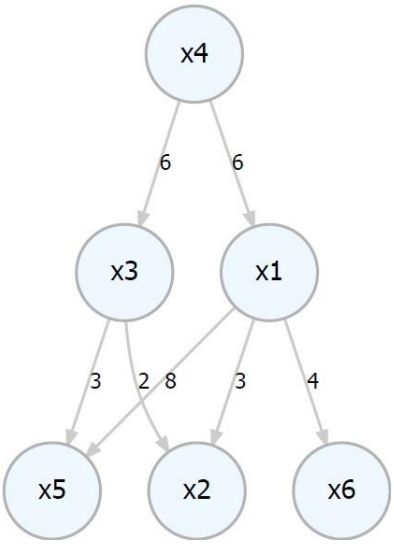
Inside the loaded source file, a function `my_simulate_example` is to simulate a data with six observed variables, from `x1` to `x6`. Function `my_simulate_example` has four arguments:

`Sample Size`	A single number that specifies the sample size (Default: `200`).
`Nonnormality`	A character that specifies the level of data non-normality ("`None`", "`Mild`", "`Median`" or "`Severe`") (Default: "`None`").
`Missingness`	A character that specifies the type of data missingness ("`None`", "`MCAR`", "`MAR`" or "`MNAR`") (Default: "`None`").
`Prop`	A single number that declares the proportion of missing data (Default: `0.10`).

Using function my_simulate_example, the simulation of datasets is based on a (6 × 6) matrix of regression coefficients (denoted by **Γ**) which looks like this:

$$\mathbf{\Gamma} = \begin{pmatrix} 0.00 & 0.00 & 0.00 & 6.00 & 0.00 & 0.00 \\ 3.00 & 0.00 & 2.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 6.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 8.00 & 0.00 & 3.00 & 0.00 & 0.00 & 0.00 \\ 4.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \end{pmatrix}$$

And the corresponding path diagram is:



With this function, we can simulate an artificial data for the demonstration by typing the following command:

```
simdat <- my_simulate_example()
```

Without setting values in the arguments of function `my_simulate_example`, R assumes we endorse the default values. That means, the R object `simdat` we have just created is a data frame with the dimension of 200 by 6, as the sample size defaults to 200 and the number of observed variables is set to six according to function `my_simulate_example`.

A15.2 Causal Discovery with LiNGAM Algorithms for Observed Variables

This section is about how to discover a causal model with ICA-LiNGAM algorithm for observed variables. First, we create a new R object named `model_ICA` from R6 class `ICALiNGAM` by calling method `new()`:

```
model_ICA <- ICALiNGAM$new()
```

By doing so, we built a R object `model_ICA` based on the template of R6 class `ICALiNGAM`. At this stage, the object `model_ICA` has nothing in it. We input the dataset simdat we just simulated to `model_ICA` by typing the command:

```
model_ICA$fit(X = simdat)
```

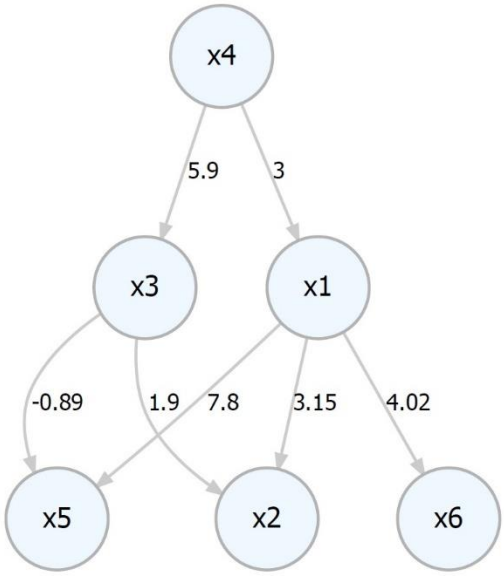
where method `fit()` is to initiate the ICA-LiNGAM algorithm; the argument `X` is the place in which we specify the dataset. Once we have typed in the command, it takes a while for R the algorithm to find out a causal model that fits the simulated data the most. In the meantime, you may find R not being responsive as the algorithm is running in the background. Once the cursor resumes blinking again, it means the task of discovering the causal model is over and we may now request the causal order of the observed variables by typing the following command:

```
model_ICA$causal_order
[1] 4 1 3 2 5 6
```

Instead of the names of the observed variables (e.g., `x1`), it shows the index of the observed variables in an ascending causal order. As one can see from the causal order, the fourth observed variable, the equivalent of the one in the fourth column of dataset `simdat` which is `x4`, is the first variable of all. And the sixth observed variable (i.e. `x6`) is the last variable of all. After that, we want to request the regression coefficient matrix of the discovered path model which can be achieved by calling field `adjacency_matrix`:

```
round(model_ICA$adjacency_matrix, digits = 3)
     [,1] [,2]   [,3]  [,4] [,5] [,6]
[1,] 0.000    0  0.000 3.002    0    0
[2,] 3.147    0  1.902 0.000    0    0
[3,] 0.000    0  0.000 5.896    0    0
[4,] 0.000    0  0.000 0.000    0    0
[5,] 7.800    0 -0.885 0.000    0    0
[6,] 4.023    0  0.000 0.000    0    0
```

To get a better view of the matrix, we make the values of elements in the matrix rounded to three decimal places. The path diagram of the resultant path model look likes:



If we want to switch the algorithm from ICA-LiNGAM to DirectLiNGAM, we create another R object named `model_DIR` from R6 class `DirectLiNGAM`:

```
model_DIR <- DirectLiNGAM$new()
```

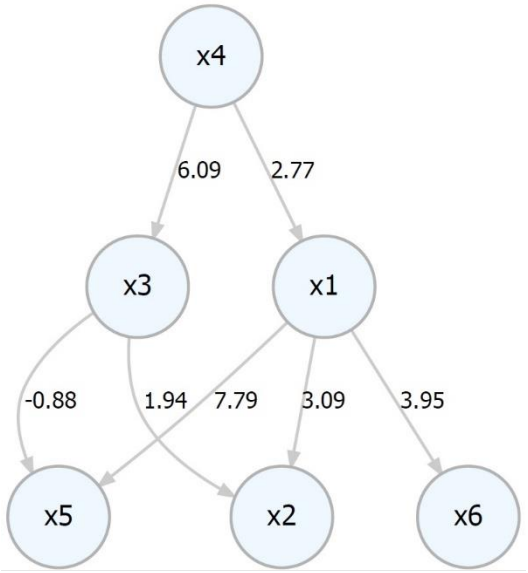
We provide the R object `model_DIR` with the dataset simdat:

```
model_DIR$fit(X = simdat)
```

Once the algorithm finishes running, we can retrieve the causal order and the regression coefficient matrix of the discovered path model by DirectLiNGAM.

```
model_DIR$causal_order
round(model_DIR$adjacency_matrix, digits=3)
      [,1] [,2]   [,3]  [,4] [,5] [,6]
[1,] 0.000    0  0.000 2.767    0    0
[2,] 3.089    0  1.935 0.000    0    0
[3,] 0.000    0  0.000 6.094    0    0
[4,] 0.000    0  0.000 0.000    0    0
[5,] 7.791    0 -0.880 0.000    0    0
[6,] 3.953    0  0.000 0.000    0    0
```

The path diagram of the discovered path model by DirectLiNGAM is:



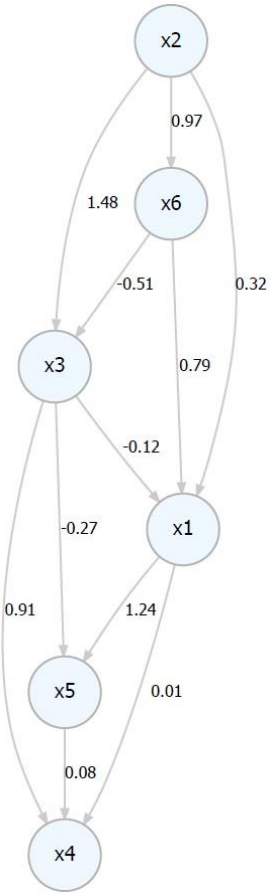
Similarly, if we want to discover a causal model this time with `ParceLiNGAM`, we create a new R object named `model_PL` from R6 class `ParceLiNGAM` by typing the following commands:

```
model_PL <- ParceLiNGAM$new()
model_PL$fit (X = simdat)
```

And we can obtain the causal order and the regression coefficient matrix of the discovered path model by ParceLiNGAM:

```
model_PL$causal_order
round(model_PL$adjacency_matrix, digits = 3)
      [,1]  [,2]   [,3] [,4]  [,5]   [,6]
[1,] 0.000 0.324 -0.119    0 0.000  0.790
[2,] 0.000 0.000  0.000    0 0.000  0.000
[3,] 0.000 1.477  0.000    0 0.000 -0.507
[4,] 0.006 0.000  0.908    0 0.076  0.000
[5,] 1.244 0.000 -0.271    0 0.000  0.000
[6,] 0.000 0.972  0.000    0 0.000  0.000
```

The resultant path diagram is:



A15.3 Causal Discovery with LiNGAM Algorithms for Latent Variables

What we have gone through are LiNGAM algorithms for observed variables. To demonstrate their variants for latent variables, we need to prepare a separate dataset. Here we will borrow population model 1 (PM1) from the manuscript for data simulation. Population model contains 15 observed variables. Function `MySimulation_PM1` is responsible for the data simulation which has four arguments:

`Sample Size`	A single number that specifies the sample size (Default: `200`).
`Nonnormality`	A character that specifies the level of data non-normality ("`None`", "`Mild`", "`Median`" or "`Severe`") (Default: "`None`").
`Missingness`	A character that specifies the type of data missingness ("`None`", "`MCAR`", "`MAR`" or "`MNAR`") (Default: "`None`").
`Prop`	A single number that declares the proportion of missing data (Default: `0.10`).

With function `MySimulate_PM1`, we simulate a dataset named `simdat` (again to overwrite the old one) by typing the following command:

```
simdat <- MySimulate_PM1()
```

The dataset simdat is a data frame with the dimension of 200 by 15, as the number of observed variables is set to 15 in PM1. A prerequisite of conducting LiNGAM-LV algorithms is a correctly specified measurement model. With respect to the dataset `simdat`, the target measurement model is a three-factor model with four cross-factor loadings (see Figure 8). Let us call the three latent variables `F1`, `F2` and `F3`. `F1` is regressed on observed variables `x1` to `x5`, `F2` is regressed on `x6` to `x10`, and `F3` is regressed on `x11` to `x15`. In addition, we have four cross-factor loadings, `F1` on `x11`, `F2` on `x1`, `F3` on `x5` and `F3` on `x10`. We will fit the said measurement model with confirmatory factor analysis using a R package of `lavaan` (Rosseel, 2012), a R package popular for latent-variable modelling. The measurement model can be expressed in `lavaan` (Rosseel, 2012) language as:

```
syntax_complete <- '
  F1 =~ x1 + x2 + x3 + x4 + x5
  F2 =~ x6 + x7 + x8 + x9 + x10
  F3 =~ x11 + x12 + x13 + x14 + x15
  F1 =~ x11
  F2 =~ x1
  F3 =~ x5 + x10
  '
```

Fitting the measurement model with function sem in lavaan can be done by:

```
cfa_out <- lavaan::sem(model = syntax_complete, data = simdat)
```

where the first argument `model` is to specify the measurement model and the second argument `data` is to specify the input dataset. What LiNGAM-LV algorithms require from the results of confirmatory factor analysis are the factor loading matrix and the factor covariance matrix. We can retrieve the (15 × 3) factor loading matrix as an R object `Lambda`:

```
Lambda <- lavaan::lavInspect(cfa_out, what = "std")$lambda
```

which is a (15 × 3) matrix. And we can obtain the (3 × 3) factor covariance matrix as an R object `Phi`:

```
Phi <- lavaan::lavInspect(cfa_out, what = "std")$psi
```

Having the measurement model for the dataset `simdat` been ready, we create a new R object named `model_ICA_LV` from R6 class `ICALiNGAM_LV` by calling method `new()`:

```
model_ICA_LV <- ICALiNGAM_LV$new()
```

Prior to discovering the causal model, the object `model_ICA_LV` requires three inputs in addition to the dataset `simdat`. The first input is the pure measurement model. A pure measurement model refers to the one that observed variables are associated with one and only one latent variable, with no cross-factor loadings

allowed. We can express the pure form of the measurement model of dataset simdat (named as `syntax_pure`) in lavaan language as:

```
syntax_pure <- c(
    "F1 =~ x1 + x2 + x3 + x4 + x5",
    "F2 =~ x6 + x7 + x8 + x9 + x10",
    "F3 =~ x11 + x12 + x13 + x14 + x15")
```

The R object `syntax_pure` is a vector of characters with three elements in it. The first element describes the relationship between latent variable `F1` and its observed indicators of `x1` to `x5` (i.e., `F1 =~ x1 + x2 + x3 + x4 + x5`) . The second element describes the relationship between `F2` and its observed indicators (i.e., `F2 ~= x6 + x7 + x8 + x9 + x10`). The third element is about `F3` (i.e., `F3 ~= x11 + x12 + x13 + x14 + x15`). We can fill in the field `model_pure` in `model_ICA_LV` to specify the pure measurement model in ICA-LiNGAM-LV by typing this command:

```
model_ICA_LV$model_pure <- syntax_pure
```

The second input is the remaining specification of a complete target measurement model in `syntax_complete` that is not declared in `syntax_pure`. From PM1 we are aware that four cross-factor loadings are not mentioned in `syntax_pure`. The four cross-factor loadings can be expressed in lavaan language as a R object `syntax_other`:

```
syntax_other <- '
  F1 =~ x11
  F2 =~ x1
  F3 =~ x5 + x10'
```

R object `syntax_other` is fed into field `model_other` in `model_ICA_LV` to specify the cross-factor loadings:

```
model_ICA_LV$model_other <- syntax_other
```

The third and final input is the factor loading matrix which is stored in R object `Lambda`. We enter `Lambda` to field `lambda` in `model_ICA_LV` to specify the factor loading matrix in ICA-LiNGAM-LV:

```
model_ICA_LV$lambda <- Lambda
```

Once all three inputs are in place, we can run ICA-LiNGAM-LV by calling method `fit()` in `model_ICA_LV`:

```
model_ICA_LV$fit(X = simdat)
```

When the estimation of ICA-LiNGAM-LV is over, we can request the causal order of the three latent variables from F1 to F3:

```
model_ICA_LV$causal_order
```

```
[1] 2 1 3
```

And the structural coefficient matrix of the discovered path model with latent variables by ICA-LiNGAM-LV:

```
round(model_ICA_LV$adjacency_matrix, digits=3)
      [,1]  [,2] [,3]
[1,] 0.000 0.685    0
[2,] 0.000 0.000    0
[3,] 0.465 0.000    0
```

The diagram of the discovered path model by ICA-LiNGAM-LV is:

Turning to DirectLiNGAM-LV, we create a new R object named `model_DIR_LV` from R6 class `DirectLiNGAM_LV` by calling method `new()`:

```
model_DIR_LV <- DirectLiNGAM_LV$new()
```

In addition to three inputs as required in `model_ICA_LV`, `model_DIR_LV` needs one more input which is the factor covariance matrix for factor scores approximation. We enter R object `Phi` to field `phi` in `model_DIR_LV` to specify the factor covariance matrix as obtained from the confirmatory factor analysis we ran earlier.

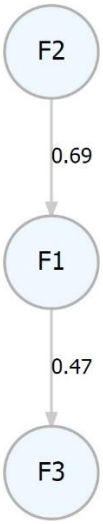
```
model_DIR_LV$model_pure <- syntax_pure
model_DIR_LV$model_other <- syntax_other
model_DIR_LV$lambda <- Lambda
model_DIR_LV$phi <- Phi
```

With all inputs ready, it is time to run DirectLiNGAM-LV by calling method `fit()` method and return the results of the causal order as well as the structural coefficient matrix of the discovered path model by DirectLiNGAM-LV:

```
model_DIR_LV$fit(X = simdat)
model_DIR_LV$causal_order
```

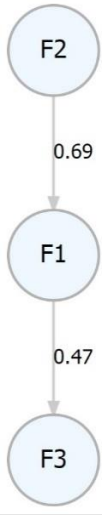
```
[1] 2 1 3
round(model_DIR_LV$adjacency_matrix, digits=3)
      [,1]  [,2] [,3]
[1,] 0.000 0.685    0
[2,] 0.000 0.000    0
[3,] 0.465 0.000    0
```

whose path diagram looks like:



In the same vein, we can change the algorithm to ParceLiNGAM-LV by typing the following commands:

```
model_PL_LV <- ParceLiNGAM_LV$new()
model_PL_LV$model_pure <- syntax_pure
model_PL_LV$model_other <- syntax_other
model_PL_LV$lambda <- Lambda
model_PL_LV$phi <- Phi
model_PL_LV$fit(X = simdat)
model_PL_LV$causal_order
[1] 2 3 1
round(model_PL_LV$adjacency_matrix, digits=3)
      [,1]  [,2]  [,3]
 [1,]    0 0.685 0.000
[2,]    0 0.000 0.000
 [3,]    0 0.465 0.000
```



Reference

Finch, & French, B. F. (2015). *Latent variable modeling with R*. Routledge.

Appendix 16: R code for LiNGAM algorithms

```
# R Function of LiNGAM_LV V01

if (!require("tidyverse")) install.packages("tidyverse")
if (!require("missMDA")) install.packages("missMDA")
if (!require("fastICA")) install.packages("fastICA")
if (!require("ica")) install.packages("ica")
if (!require("clue")) install.packages("clue")
if (!require("lavaan")) install.packages("lavaan")
if (!require("regsem")) install.packages("regsem")
if (!require("psych")) install.packages("psych")
if (!require("PearsonDS")) install.packages("PearsonDS")
if (!require("lars")) install.packages("lars")
if (!require("glmnet")) install.packages("glmnet")
if (!require("mice")) install.packages("mice")
if (!require("data.table")) install.packages("data.table")
if (!require("mnonr")) install.packages("mnonr")


require(tidyverse)
require(missMDA)
require(fastICA)
require(ica)
require(clue)
require(lavaan)
require(regsem)
require(psych)
require(PearsonDS)
require(lars)
require(glmnet)
require(mice)
require(data.table)
require(mnonr)

#---- ICA-LiNGAM_LV ----


ICALiNGAM_LV <- R6::R6Class(
  "ICALiNGAM_LV",
  public = list(

    #' @title ICALiNGAM_LV class
    #' @description
    #' To discover the causal relations between latent factors using ICA-
LiNGAM algorithm for latent variables
    #' @param lambda A matrix that specifies a factor loading matrix or the
relations between p observed variables and q latent variables
    #' @param model_pure A vector of characters that specifies lavaan model
syntax for pure measurement model
    #' @param model_other A character of other specification of measurement
model (e.g., cross-loading)
    #' @param max_k a numeric that specifies the maximum attempt to search
causal order (Default: 3)
    #' @param max_iter_ica a single numeric that specifies the maximum
iteration of ICA (Default: 1000)
    #' @param type_ica a character that assigns R package between "fastICA"
and "ica" to run ICA (Default: "fastICA")
    #' @param regsem_penalty a character that specifies the penalty type in
regsem::cv_regsem function (Default: alasso)
```

```
    #' @param regsem_nlambda a single numeric that specifies the penalty
value to test in regsem::cvregsem function (Default: 20)
    #' @param regsem_jump a single numeric that specifies the change in
penalty values per iteration in regsem::cvregsem function (Default: .03)


    lambda = NULL,
    model_pure = NULL,
    model_other = NULL,
    max_k = 3,
    max_iter_ica = 1000,
    type_ica = "fastICA",
    regsem_penalty = "alasso",
    regsem_nlambda = 20,
    regsem_jump = 0.03,


    causal_order = NULL,
    adjacency_matrix = NULL,
    listOV = NULL,
    listLV = NULL,

    initialize = function(
      lambda = NULL,
      model_pure = NULL,
      model_other = NULL,
      max_k = 3,
      max_iter_ica = 1000,
      type_ica = "fastICA",
      regsem_penalty = "alasso",
      regsem_nlambda = 20,
      regsem_jump = 0.03){

      self$lambda <- lambda
      self$model_pure <- model_pure
      self$model_other <- model_other
      self$max_k <- max_k
      self$max_iter_ica <- max_iter_ica
      self$type_ica <- type_ica
      self$regsem_penalty <- regsem_penalty
      self$regsem_nlambda <- regsem_nlambda
      self$regsem_jump <- regsem_jump
    },

    fit = function(X){

      # Specify the number of latent variables
      q <- dim(self$lambda)[2]

      # Name latent variables and observed variables
      self$name_variables()

      # Impute missing values
      if (length(which(is.na(X))) >= 1){
        X <- missMDA::imputePCA(X, ncp = q) %>%
          .$completeObs %>%
          as.data.frame()
      }

      history_k <- list()
```

```
while(TRUE){
for (i_k in c(1:self$max_k)){

  # Estimate A_ica from ICA

  # Summon fastICA from fastICA Package
  if(tolower(self$type_ica) == "fastica"){
    ica_out <- X %>% fastICA::fastICA(
      n.comp = q,
      alg.typ = "parallel",
      maxit = self$max_iter_ica,
      fun = "logcosh")
    A_ica <- (ica_out$K %*% ica_out$W)}


  # Summon fastICA from ica package
  if(tolower(self$type_ica) == "icafast"){
    ica_out <- X %>% ica::icafast(
      nc = q,
      alg = "par",
      maxit = self$max_iter_ica,
      fun = "logcosh")
    A_ica <- (t(ica_out$Q) %*% ica_out$R)}


  # Summon information maximum from ica Package
  if(tolower(self$type_ica) == "icaimax"){
    ica_out <- X %>% ica::icaimax(
      nc = q,
      maxit = self$max_iter_ica,
      alg = "newton",
      fun = "tanh")
    A_ica <- (t(ica_out$Q) %*% ica_out$R)}


  # Summon JADE from ica Package
  if(tolower(self$type_ica) == "icajade"){
    ica_out <- X %>% ica::icajade(
      nc = q,
      maxit = self$max_iter_ica)
    A_ica <- (t(ica_out$Q) %*% ica_out$R)}


  # Log the variable name to A_ica from X
  rownames(A_ica) <- colnames(X)


  # Retrieve unmixing matrix W_ica
  lambda <- as.matrix(self$lambda)
  W_ica <- (solve(t(lambda) %*% A_ica)) %*% (t(lambda) %*% lambda)


  # Permute W_ica
  linassign <- clue::solve_LSAP(1 / abs(t(W_ica)), maximum = FALSE)
  row_index <- unname(linassign)
  PW_ica    <- t(W_ica[row_index,])


  # Scaling vector is the diagonal of permuted W_ica
  D <- diag(PW_ica)
```

```
      # Estimate adjacency matrix
      B_ica <- diag(q) - (PW_ica / D)
      colnames(B_ica) <- rownames(B_ica)


      # Log the estimated causal order
      history_k[[i_k]] <- self$estimate_causal_order(B_ica)
    }

    selected_k <- findListMax(history_k)

    # If no ties happen, endorse the causal order as the finalized
    if(all(is.na(selected_k)) == FALSE){
        self$causal_order <- selected_k
        break
    }
    }

    # Estimate a preliminary structural matrix
    B_preliminary <- X %>% self$estimate_structural_matrix()


    # Finalize the structural matrix
    self$adjacency_matrix <- X %>% self$my_regsem_final(B_preliminary)
  },


  name_variables = function(){


    # Specify the number of latent variables
    q <- dim(self$lambda)[2]


    # Identify latent variables and their corresponding observed
variables
    listLV <- list(); listOV <- list()


    for (i_LV in c(1:q)){

      # Find out the latent variable label
      tmp_LV <- gsub("=~.*","",self$model_pure[i_LV])


      # Remove spacing from the label
      self$listLV[[i_LV]] <- gsub(" ", "", tmp_LV, fixed = TRUE) %>%
        as.character()


      # Find out the corresponding observed variables
      tmp_OVs <- gsub(".*=~","",self$model_pure[i_LV])


      # And convert it into a vector
      tmp_OVs <- gsub(" ", "", tmp_OVs, fixed = TRUE)
      self$listOV[[i_LV]] <- unlist(strsplit(tmp_OVs,"\\+"))

    }
  },
```

```
estimate_causal_order = function(B) {
  d <- ncol(B)

  # Set m(m + 2) smallest elements (in absolute value) of B to zero
  pos_vec <- order(abs(B))
  initial_zero_num <- d * (d + 1) / 2
  for (i in pos_vec[1:initial_zero_num]) {B[i] <- 0}


  # Set to zero until DAG is obtained
  for (i in pos_vec[(initial_zero_num + 1):length(pos_vec)]) {
    B[i] <- 0

    # if B is not DAG, return null
    causal_order <- self$search_causal_order(B)
    if (!is.null(causal_order)) {
      break
    }
  }
  return(causal_order)
},


search_causal_order = function(B) {
  causal_order   <- c()
  q              <- nrow(B)
  original_index <- c(1:q)

  while (0 < ncol(B)) {

    # Find a row whose elements are all zero
    row_index_list <- which(rowSums(B) == 0)
    if (length(row_index_list) == 0) break


    # Append ith to the end
    target_index   <- row_index_list[1]
    causal_order   <- c(causal_order, original_index[target_index])
    original_index <- original_index[-target_index]


    # Remove ith row and ith column
    B <- as.matrix(B[-target_index, -target_index])
  }
  if (!length(causal_order) == q) {
    causal_order <- NULL
  }
  return(causal_order)
},


estimate_structural_matrix = function(X){

  if (is.null(self$pk) == FALSE){
    tmp_pk <- self$pk
    diag(tmp_pk) <- 0
  }

  q <- dim(self$lambda)[2]
```

```
      B <- matrix(0, nrow = q, ncol = q)

      for (i in 2:length(self$causal_order)){
        target <- unlist(self$causal_order[i])
        predictors <- unlist(self$causal_order[1:(i-1)])

        # Exclude variables specified in no_path with prior knowledge
        if(!is.null(self$pk)){
          predictors_pk <- c()
          for (p in predictors){
            if (tmp_pk[target, p] != 0){predictors_pk <- c(predictors_pk,
p)}
          }
          predictors <- predictors_pk
        }

        if(length(predictors) > 0){
          B[target, predictors] <- self$my_regsem(X,predictors,target)
        }
      }
      return(B)
    },


    my_regsem = function(X, predictors, target){


      # Prepare syntax for measurement model (MM)
      syntax_MM <- paste(self$model_pure[c(predictors,target)],collapse =
"\n")


      # Prepare syntax for structural model (SM)
      list_xj <- unlist(self$listLV[c(predictors)])
      xi <- unlist(self$listLV[c(target)])

      syntax_SM <- ""; count <- 0
      for (xj in list_xj){
        count <- count + 1
        tP <- paste(xi," ~ c",count,"*",xj,";\n",sep="")
        syntax_SM <- paste(syntax_SM,tP,sep="")
      }

      # Combine MM with SM
      syntax_SEM <- paste(syntax_MM,syntax_SM,sep="\n")


      # Subset dataset
      X_ <- X %>%
        subset(select = c(unlist(self$listOV[c(predictors,target)])))


      # Run a standard SEM
      sem_out <- lavaan::sem(syntax_SEM,X_,fixed.x = FALSE)


      # If there is more than one predictor, run a regularized SEM
      if (length(predictors) >=2) {
        regsem_out <- regsem::cv_regsem(
          sem_out,
```

```
          n.lambda = self$regsem_nlambda,
          type = self$regsem_penalty,
          jump = self$regsem_jump,
          max.iter = 100,
          step = 0.3,
          pars_pen = c(paste("c",c(1:count),sep="")),
          verbose = FALSE)


        # Extract all parameter estimates (pe) after penalizations
        pe <- regsem_out$final_pars


        # Select parameter estimates
        pe_selected <- c()
        for (i in c(1:length(predictors))){
          pe_selected[i] <- pe[paste(list_xj[i], " -> ", xi,sep="")]
        }

      } else {

        # If there is only one predictor, no need to run RegSEM but simply
obtain pe from SEM
        pe <- lavaan::parameterEstimates(sem_out)
        tmp_row <- which(pe$lhs == xi & pe$op == "~" & pe$rhs ==
list_xj[1])
        pe_selected <- pe[tmp_row,"est"]
      }

      return(matrix(pe_selected, ncol = length(predictors)))
    },


    my_regsem_final = function(X, B){


      # Run an overall RegSEM
      listLV <- self$listLV
      B_ <- B
      B_[B_ > 0.00] <- 1.00
      num_para <- length(B_[B_ == 1])


      # Draft syntax for structural model (SM)
      syntax_SM <- ""; count <- 0;
      for (i in c(1:length(listLV))){
        for (j in c(setdiff(c(1:length(listLV)),i))){
          if(B_[i,j] == 1){
            count <- count + 1
            ts <- paste(listLV[[i]], " ~
c",count,"*",listLV[[j]],";",sep="")
            syntax_SM <- paste(syntax_SM, ts, sep = "\n")
          }
        }
      }


      # Draft syntax for measurement model (MM)
      if (!is.null(self$model_other)){
        syntax_MM <- paste(
          stringr::str_flatten(self$model_pure, collapse = "\n"),
```

```
          self$model_other, collapse = "\n")
      } else {
        syntax_MM <- stringr::str_flatten(self$model_pure, collapse = "\n")
      }


      # Combine model syntax
      syntax_SEM <- paste(syntax_SM, syntax_MM, sep = "\n")


      # Run a standard SEM
      sem_out <- lavaan::sem(syntax_SEM, X)


      # Run a regularized SEM
      regsem_out <- regsem::cv_regsem(
        sem_out,
        n.lambda = self$regsem_nlambda,
        type = self$regsem_penalty,
        jump = self$regsem_jump,
        max.iter = 100,
        step = 0.3,
        pars_pen = c(paste("c",c(1:num_para),sep="")),
        verbose = FALSE)


      # Retrieve all parameter estimates (pe)
      pe <- regsem_out$final_pars[1:num_para]


      # Substantiate a null matrix
      beta <- matrix(0, nrow = length(listLV), ncol = length(listLV))


      # Fill in the null matrix
      for (i in c(1:length(listLV))){
        for (j in c(setdiff(c(1:length(listLV)),i))){
          if(B_[i,j] == 1){
            beta[i,j] <- pe[[paste(listLV[[j]], " -> ",
listLV[[i]],sep="")]]
          }
        }
      }

      return(beta)

    }
  )
)

findListMax <- function(mylist){
  n_u <- length(unique(mylist))
  n_l <- length(mylist)
  count_u <- c()
  for (i in c(1:n_u)){
    count <- 0
    for (j in c(1:n_l)){
      if(identical(unique(mylist)[[i]], mylist[[j]])){count = count + 1}
    }
    count_u <- c(count_u, count)
  }
```

```
  if (length(which(count_u == max(count_u)))==1){
    return(unique(mylist)[[which(count_u == max(count_u))]])
  } else {
    return(NA)
  }
}

#---- DirectLiNGAM_LV ----


DirectLiNGAM_LV <- R6::R6Class(
  "DirectLiNGAM_LV",
  public = list(

    #' @title DirectLiNGAM_LV class
    #' @description To discover the causal relations between latent factors
using DirectLiNGAM algorithm for latent variables
    #' @param lambda A matrix that specifies factor loading matrix or
relations between p observed variables and q latent variables, or factor
loading matrix
    #' @param phi A matrix that specifies covariance between q latent
variables
    #' @param model_pure A vector of characters that specifies lavaan model
syntax for pure measurement model
    #' @param model_other A character of other specification of measurement
model (e.g., cross-loading)
    #' @param regsem_penalty a character that specifies the penalty type
for regsem::cv_regsem function (Default: "alasso")
    #' @param regsem_nlambda a numeric that specifies the penalty value to
test for regsem::cvregsem function (Default: 20)
    #' @param regsem_jump a numeric that specifies the change in penalty
values for each iteration for regsem::cvregsem function (Default: .03)
    #' @param pk A squared matrix of prior knowledge, with dimension of
factors (q) by factors (q)
    #' @param fsa A vector of characters that specifies factor scores
approximation approaches (Default: "Thurstone", "tenBerge", "Bartlett")
    #' @param fsa_user A character that specifies the factor score
approximation approach to endorse if causal orders are not consistent
(Default: "tenBerge")


    lambda = NULL,
    phi = NULL,
    model_pure = NULL,
    model_other = NULL,
    regsem_penalty = "alasso",
    regsem_nlambda = 20,
    regsem_jump = 0.03,
    pk = NULL,
    fsa = c("Thurstone","tenBerge","Bartlett"),
    fsa_user = "tenBerge",


    causal_order = NULL,
    causal_order_fsa = list(),
    adjacency_matrix = NULL,
    partial_orders = NULL,
    listOV = NULL,
    listLV = NULL,
```

```
initialize = function(
  lambda = NULL,
  phi = NULL,
  model_pure = NULL,
  model_other = NULL,
  regsem_penalty = "alasso",
  regsem_nlambda = 20,
  regsem_jump = 0.03,
  pk = NULL,
  fsa = c("Thurstone","tenBerge","Bartlett"),
  fsa_user = "tenBerge"){

  self$lambda <- lambda
  self$phi <- phi
  self$model_pure <- model_pure
  self$model_other <- model_other
  self$regsem_penalty <- regsem_penalty
  self$regsem_nlambda <- regsem_nlambda
  self$regsem_jump <- regsem_jump
  self$pk <- pk
  self$fsa <- fsa
  self$fsa_user <- fsa_user
},

fit = function(X){

  # Specify the number of latent variables
  q <- dim(self$lambda)[2]


  # Check parameters
  if(is.null(self$pk) == FALSE){
    if(dim(self$pk)[1] != q |
       dim(self$pk)[2] != q){
      sprintf("The shape of prior knowledge must be (%d, %d)", q, q)
    }
    self$partial_orders <- self$extract_partial_orders(self$pk)
  }


  # Name latent variables and observed variables
  self$name_variables()


  # Check if the user-specified fsa is on the list of fsa
  if (!(self$fsa_user %in% self$fsa)){

    # If not, renew the list
    self$fsa <- c(self$fsa, self$fsa_user)
  }


  for (i_fsa in self$fsa){

    # Approximate factor scores
    X_fsa <- self$compute_factor_scores(X, i_fsa)


    # Causal discovery
    K <- self$discover_causal_order(X_fsa)
```

```
      # Log  causal order
      self$causal_order_fsa[[i_fsa]] <- K
    }


    # Check if all possible causal orders are consistent
    is_consistent <- TRUE

    for (i_fsa in self$fsa){
      for (j_fsa in setdiff(self$fsa, i_fsa)){

        k_i_fsa <- self$causal_order_fsa[[i_fsa]]
        k_j_fsa <- self$causal_order_fsa[[j_fsa]]

        if (!(identical(k_i_fsa, k_j_fsa))){
          is_consistent <- FALSE
          break
        }
      }
    }

    # If all causal orders are identical, endorse any
    if (is_consistent){
      self$causal_order <- self$causal_order_fsa[[1]]
    } else {

      # If not identical, endorse the user-specified approach
      self$causal_order <- self$causal_order_fsa[[self$fsa_user]]
    }


    # Estimate a preliminary structural matrix
    B_preliminary <- self$estimate_structural_matrix(X)


    # Finalize structural matrix
    self$adjacency_matrix <- self$my_regsem_final(X, B_preliminary)
  },


  name_variables = function(){


    # Specify the number of latent variables
    q <- dim(self$lambda)[2]


    # Identify latent variables and their corresponding observed
variables
    listLV <- list(); listOV <- list()


    for (i_LV in c(1:q)){

      # Find out the latent variable label
      tmp_LV <- gsub("=~.*","",self$model_pure[i_LV])


      # Remove spacing from the label
```

```
        self$listLV[[i_LV]] <- gsub(" ", "", tmp_LV, fixed = TRUE) %>%
          as.character()


        # Find out the corresponding observed variables
        tmp_OVs <- gsub(".*=~","",self$model_pure[i_LV])


        # And convert it into a vector
        tmp_OVs <- gsub(" ", "", tmp_OVs, fixed = TRUE)
        self$listOV[[i_LV]] <- unlist(strsplit(tmp_OVs,"\\+"))

      }
    },


    compute_factor_scores = function(X, method){

return(psych::factor.scores(X,self$lambda,self$phi,method=method,impute="me
dian")$scores)
    },


    discover_causal_order = function(X){

      # Specify the number of latent variables
      q <- dim(self$lambda)[2]

      # Number latent variables
      U <- c(1:q)

      # Substantive an empty list of causal order (K)
      K <- c()

      for (i_LV in c(1:q)){

        # Determine the first variable (m)
        m <- self$search_causal_order(X, U)

        # Remove the effect of m on the rest of U
        for (i in U){
          if (i != m){
            X[,i] <- self$residual(X[,i], X[,m])
          }
        }

        # Variable m entered the end of K
        K <- c(K, m)

        # Drop m from U
        U <- U[U != m]

        # Update partial orders
        # if (is.null(self$pk) == FALSE){
        #   self$partial_orders <-
self$partial_orders[which(self$partial_orders[,1]!=m),]
        # }
      }

      return(K)
    },
```

```
extract_partial_orders = function(pk){

  # Extract partial orders from prior knowledge.
  pos_path_pair <- which(pk == 1)
  pos_nopath_pair <- which(pk == 0)

  if (length(pos_path_pair) >= 1){
    # Check for inconsistencies in pairs with path
    path_pair <- list()
    for (i in c(1:length(pos_path_pair))){
      tIndex <- pos_path_pair[i]
      if (tIndex %% ncol(pk) == 0){
        tRow <- ncol(pk)
        tCol <- tIndex %/% ncol(pk)
      } else {
        tRow <- tIndex %% ncol(pk)
        tCol <- tIndex %/% ncol(pk) + 1
      }
      path_pair[[i]] <- c(tCol, tRow)
    }
  } else {path_pair <- list()}

  if (length(pos_nopath_pair) >= 1){
    # Check for inconsistencies in pairs without path
    nopath_pair <- list()
    for (i in c(1:length(pos_nopath_pair))){
      tIndex <- pos_nopath_pair[[i]]
      if (tIndex %% ncol(pk) == 0){
        tRow <- ncol(pk)
        tCol <- tIndex %/% ncol(pk)
      } else {
        tRow <- tIndex %% ncol(pk)
        tCol <- tIndex %/% ncol(pk) + 1
      }
      nopath_pair[[i]] <- sort(c(tCol, tRow))
    }
    nopath_pair <- unique(nopath_pair)
  } else {nopath_pair <- list()}

  all_pairs <- c(path_pair, nopath_pair)

  if (length(all_pairs)==0) {
    # if no pairs are extracted from the specified prior knowledge
    # discard the prior knowledge.
    self$pk <- NULL
    return(NULL)
  }
  out <- as.data.frame(matrix(c(unlist(all_pairs)),ncol=2,byrow=T))
  colnames(out) <- c("From","To")
  return(out)
},


residual = function(xi, xj){
  # The residual when xi is regressed on xj.
  xi - (cov(xi, xj) / var(xj)) * xj
},
```

```
entropy = function(u){

  # Calculate entropy using the maximum entropy approximation
  k1 <- 79.047
  k2 <- 7.4129
  gamma <- 0.37457
  return(
    (1 + log(2 * pi)) / 2 -
      k1 * (mean(log(cosh(u))) - gamma)^2 -
      k2 * (mean(u * exp((-u^2) / 2)))^2)
},


diff_mutual_info = function(xi_std, xj_std, ri_j, rj_i){
  # Calculate the difference of the mutual information.
  term1 <- self$entropy(xj_std) + self$entropy(ri_j / sd(ri_j))
  term2 <- self$entropy(xi_std) + self$entropy(rj_i / sd(rj_i))
  return(term1 - term2)
},


search_candidate = function(U){
  # Search for candidate features.

  # If no prior knowledge is specified, nothing to do.
  if (is.null(self$pk)){
    out <- c()
    out$Uc <- U
    out$Vj <- list()
    return(out)
  }

  # Find exogenous features
  Uc = c()
  for (j in U){
    index = U[U != j]
    if(sum(self$pk[j,index]) == 0){Uc <- c(Uc, j)}
  }

  # Find endogenous features, and then find candidate features
  if (length(Uc) == 0){
    U_end <- c()
    for (j in U){
      index <- U[U != j]
      if(sum(self$pk[j,index],na.rm=T) > 0){U_end <- c(U_end, j)}
    }

    # Find sink features (original)
    for (i in U){
      index <- U[U != i]
      if(sum(self$pk[index, i]) == 0){U_end <- c(U_end, i)}
    }
    Uc <- setdiff(U, U_end)
  }

  # make V^(j)
  Vj <- c()
  for (i in U){
    if (i %in% Uc){
      if (sum(self$pk[i,Uc]==0)){Vj <- c(Vj, i)}
    }
```

```
    }
    out <- c()
    if (is.null(Uc)){out$Uc <- U}else{out$Uc <- c(Uc)}
    out$Vj <- c(Vj)
    return(out)
  },


  search_causal_order = function(X, U){

    # Search causal order
    sc_out <- self$search_candidate(U)
    Uc <- sc_out$Uc
    Vj <- sc_out$Vj

    if (length(Uc) == 1){return(Uc[1])}

    # Create a list to log the values of M criterion
    M_list <- c()


    for (i in Uc){
      M <-0
      for (j in U){
        if (i != j){

          # Standardize variables xj and xj
          xi_std = (X[, i] - mean(X[, i])) / sd(X[, i])
          xj_std = (X[, j] - mean(X[, j])) / sd(X[, j])


          if (i %in% Vj & j %in% Uc){
            ri_j <- xi_std
          } else {
            # Compute the error in least square regression when xi is
regressed on xj
            ri_j <- self$residual(xi_std, xj_std)}


          if (j %in% Vj & i %in% Uc){
            rj_i <- xj_std
          } else {
            # Compute the error in least square regression when xj is
regressed on xj
            rj_i <- self$residual(xj_std, xi_std)}


          # Renew the value of M criterion
          M <- M + min(0,self$diff_mutual_info(xi_std, xj_std, ri_j,
rj_i))^2
        }
      }

      # Log the M criterion for variable i
      M_list <- c(M_list, -M)
    }
    return(Uc[which.max(M_list)])
  },


  estimate_structural_matrix = function(X){
```

```
      if (is.null(self$pk) == FALSE){
        tmp_pk <- self$pk
        diag(tmp_pk) <- 0
      }

      q <- dim(self$lambda)[2]

      B <- matrix(0, nrow = q, ncol = q)

      for (i in 2:length(self$causal_order)){
        target <- unlist(self$causal_order[i])
        predictors <- unlist(self$causal_order[1:(i-1)])

        # Exclude variables specified in no_path with prior knowledge
        if(!is.null(self$pk)){
          predictors_pk <- c()
          for (p in predictors){
            if (tmp_pk[target, p] != 0){predictors_pk <- c(predictors_pk,
p)}
          }
          predictors <- predictors_pk
        }

        if(length(predictors) > 0){
          B[target, predictors] <- self$my_regsem(X,predictors,target)
        }
      }
      return(B)
    },


    my_regsem = function(X, predictors, target){


      # Prepare syntax for measurement model (MM)
      syntax_MM <- paste(self$model_pure[c(predictors,target)],collapse =
"\n")


      # Prepare syntax for structural model (SM)
      list_xj <- unlist(self$listLV[c(predictors)])
      xi <- unlist(self$listLV[c(target)])

      syntax_SM <- ""; count <- 0
      for (xj in list_xj){
        count <- count + 1
        tP <- paste(xi," ~ c",count,"*",xj,";\n",sep="")
        syntax_SM <- paste(syntax_SM,tP,sep="")
      }

      # Combine MM with SM
      syntax_SEM <- paste(syntax_MM,syntax_SM,sep="\n")


      # Subset dataset
      X_ <- X %>%
        subset(select = c(unlist(self$listOV[c(predictors,target)])))


      # Run a standard SEM
```

```
      sem_out <- lavaan::sem(syntax_SEM,X_,fixed.x = FALSE)


      # If there is more than one predictor, run a regularized SEM
      if (length(predictors) >=2) {
        regsem_out <- regsem::cv_regsem(
          sem_out,
          n.lambda = self$regsem_nlambda,
          type = self$regsem_penalty,
          jump = self$regsem_jump,
          max.iter = 100,
          step = 0.3,
          pars_pen = c(paste("c",c(1:count),sep="")),
          verbose = FALSE)


        # Extract all parameter estimates (pe) after penalizations
        pe <- regsem_out$final_pars


        # Select parameter estimates
        pe_selected <- c()
        for (i in c(1:length(predictors))){
          pe_selected[i] <- pe[paste(list_xj[i], " -> ", xi,sep="")]
        }

      } else {

        # If there is only one predictor, no need to run RegSEM but simply
obtain pe from SEM
        pe <- lavaan::parameterEstimates(sem_out)
        tmp_row <- which(pe$lhs == xi & pe$op == "~" & pe$rhs ==
list_xj[1])
        pe_selected <- pe[tmp_row,"est"]
      }

      return(matrix(pe_selected, ncol = length(predictors)))
    },


    my_regsem_final = function(X, B){


      # Run an overall RegSEM
      listLV <- self$listLV
      B_ <- B
      B_[B_ > 0.00] <- 1.00
      num_para <- length(B_[B_ == 1])


      # Draft syntax for structural model (SM)
      syntax_SM <- ""; count <- 0;
      for (i in c(1:length(listLV))){
        for (j in c(setdiff(c(1:length(listLV)),i))){
          if(B_[i,j] == 1){
            count <- count + 1
            ts <- paste(listLV[[i]], " ~
c",count,"*",listLV[[j]],";",sep="")
            syntax_SM <- paste(syntax_SM, ts, sep = "\n")
          }
        }
```

```
      }


      # Draft syntax for measurement model (MM)
      if (!is.null(self$model_other)){
        syntax_MM <- paste(
          stringr::str_flatten(self$model_pure, collapse = "\n"),
          self$model_other, collapse = "\n")
      } else {
        syntax_MM <- stringr::str_flatten(self$model_pure, collapse = "\n")
      }


      # Combine model syntax
      syntax_SEM <- paste(syntax_SM, syntax_MM, sep = "\n")


      # Run a standard SEM
      sem_out <- lavaan::sem(syntax_SEM, X)


      # Run a regularized SEM
      regsem_out <- regsem::cv_regsem(
        sem_out,
        n.lambda = self$regsem_nlambda,
        type = self$regsem_penalty,
        jump = self$regsem_jump,
        max.iter = 100,
        step = 0.3,
        pars_pen = c(paste("c",c(1:num_para),sep="")),
        verbose = FALSE)


      # Retrieve all parameter estimates (pe)
      pe <- regsem_out$final_pars[1:num_para]


      # Substantiate a null matrix
      beta <- matrix(0, nrow = length(listLV), ncol = length(listLV))


      # Fill in the null matrix
      for (i in c(1:length(listLV))){
        for (j in c(setdiff(c(1:length(listLV)),i))){
          if(B_[i,j] == 1){
            beta[i,j] <- pe[[paste(listLV[[j]], " -> ",
listLV[[i]],sep="")]]
          }
        }
      }

      return(beta)

    }
  )
)

#---- ParceLiNGAM_LV ----


ParceLiNGAM_LV <- R6::R6Class(
```

```
  "ParceLiNGAM_LV",
  public = list(

    #' @title ParceLiNGAM_LV class
    #' @description To discover the causal relations between latent factors
using ParceLiNGAM algorithm for latent variables
    #' @param lambda A matrix of relations between p observed variables and
q latent variables, or factor loading matrix
    #' @param phi A matrix of relations between q latent variables
    #' @param phi A matrix of relations between q latent variables
    #' @param model_pure A vector of characters that specifies lavaan model
syntax for pure measurement model
    #' @param model_other A character of other specification of measurement
model (e.g., cross-loading)
    #' @param regsem_penalty a character that specifies the penalty type
for regsem::cv_regsem function (Default: alasso)
    #' @param regsem_nlambda a numeric that specifies the penalty value to
test for regsem::cvregsem function (Default: 20)
    #' @param regsem_jump a numeric that specifies the change in penalty
values for each iteration for regsem::cvregsem function (Default: .03)
    #' @param pk A squared matrix of prior knowledge, with dimension of
factors (q) by factors (q)
    #' @param fsa A vector of characters that specifies factor scores
approximation approaches (Default: Thurstone, tenBerge, Bartlett)
    #' @param fsa_user A character that specifies the factor score
approximation approach to endorse if causal orders are not consistent
(Default: tenBerge)
    #' @param alpha a numeric that specifies a uncorrected threshold value
for Fisher's independence test (Default: .001)


    lambda = NULL,
    phi = NULL,
    model_pure = NULL,
    model_other = NULL,
    regsem_penalty = "alasso",
    regsem_nlambda = 20,
    regsem_jump = 0.03,
    pk = NULL,
    fsa = c("Thurstone","tenBerge","Bartlett"),
    fsa_user = "tenBerge",
    alpha = .001,


    causal_order = NULL,
    causal_order_fsa = list(),
    adjacency_matrix = NULL,
    listOV = NULL,
    listLV = NULL,
    list_pHSIC = NULL,
    list_pHSIC_fsa = list(),


    initialize = function(
      lambda = NULL,
      phi = NULL,
      model_pure = NULL,
      model_other = NULL,
      regsem_penalty = "alasso",
      regsem_nlambda = 20,
      regsem_jump = 0.03,
```

```
  pk = NULL,
  fsa = c("Thurstone","tenBerge","Bartlett"),
  fsa_user = "tenBerge",
  alpha = .001){


  self$lambda <- lambda
  self$phi <- phi
  self$model_pure <- model_pure
  self$model_other <- model_other
  self$regsem_penalty <- regsem_penalty
  self$regsem_nlambda <- regsem_nlambda
  self$regsem_jump <- regsem_jump
  self$pk <- pk
  self$fsa <- fsa
  self$fsa_user <- fsa_user
  self$alpha <- alpha
},

fit = function(X){

  # Specify the number of latent variables
  q <- dim(self$lambda)[2]


  # Check prior knowledge
  if(is.null(self$pk) == FALSE){
    if(dim(self$pk)[1] != q |
       dim(self$pk)[2] != q){
      sprintf("The shape of prior knowledge must be (%d, %d)", q, q)
    }
    self$partial_orders <- self$extract_partial_orders(self$pk)
  }


  if (self$alpha < 0){cat("alpha must be an float greater than 0.\n")}


  # Name latent variables and observed variables
  self$name_variables()


  # Check if the user-specified fsa is on the list of fsa
  if (!(self$fsa_user %in% self$fsa)){

    # If not, renew the list
    self$fsa <- c(self$fsa, self$fsa_user)
  }


  for (i_fsa in self$fsa){

    # Approximate factor scores & center
    X_fsa <- X %>%
      self$compute_factor_scores(i_fsa) %>%
      scale()


    # Discover causal order
    causal_order_out <- X_fsa %>% self$discover_causal_order()
```

```
      # Log causal order
      self$causal_order_fsa[[i_fsa]] <- causal_order_out$K
      self$list_pHSIC_fsa[[i_fsa]] <- causal_order_out$p_bttm

    }

    # Check if all possible causal orders are consistent
    is_consistent <- TRUE

    for (i_fsa in self$fsa){
      for (j_fsa in setdiff(self$fsa, i_fsa)){
        k_i_fsa <- self$causal_order_fsa[[i_fsa]]
        k_j_fsa <- self$causal_order_fsa[[j_fsa]]

        if (!(identical(k_i_fsa, k_j_fsa))){
          is_consistent <- FALSE
          break
        }
      }
    }

    # If all causal orders are identical endorse any
    if (is_consistent){
      self$causal_order <- self$causal_order_fsa[[1]]
      self$list_pHSIC <- self$list_pHSIC_fsa[[1]]
    } else {

      # If not identical, endorse the user-specified approach
      self$causal_order <- self$causal_order_fsa[[self$fsa_user]]
      self$list_pHSIC <- self$list_pHSIC_fsa[[self$fsa_user]]
    }

    # Estimate a preliminary structural matrix
    B_preliminary <- self$estimate_structural_matrix(X)


    # Finalize structural matrix
    self$adjacency_matrix <- X %>% self$my_regsem_final(B_preliminary)
  },

  name_variables = function(){


    # Specify the number of latent variables
    q <- dim(self$lambda)[2]


    # Identify latent variables and their corresponding observed
variables
    listLV <- list(); listOV <- list()


    for (i_LV in c(1:q)){

      # Find out the latent variable label
      tmp_LV <- gsub("=~.*","",self$model_pure[i_LV])


      # Remove spacing from the label
      self$listLV[[i_LV]] <- gsub(" ", "", tmp_LV, fixed = TRUE) %>%
```

```
      as.character()


    # Find out the corresponding observed variables
    tmp_OVs <- gsub(".*=~","",self$model_pure[i_LV])


    # And convert it into a vector
    tmp_OVs <- gsub(" ", "", tmp_OVs, fixed = TRUE)
    self$listOV[[i_LV]] <- unlist(strsplit(tmp_OVs,"\\+"))

  }
},


compute_factor_scores = function(X, method){

return(psych::factor.scores(X,self$lambda,self$phi,method=method,impute="me
dian")$scores)
},


extract_partial_orders = function(pk){

  # Extract partial orders from prior knowledge.
  pos_path_pair <- which(pk == 1)
  pos_nopath_pair <- which(pk == 0)

  if (length(pos_path_pair) >= 1){
  # Check for inconsistencies in pairs with path
  path_pair <- list()
    for (i in c(1:length(pos_path_pair))){
      tIndex <- pos_path_pair[i]
      if (tIndex %% ncol(pk) == 0){
        tRow <- ncol(pk)
        tCol <- tIndex %/% ncol(pk)
      } else {
        tRow <- tIndex %% ncol(pk)
        tCol <- tIndex %/% ncol(pk) + 1
      }
      path_pair[[i]] <- c(tCol, tRow)
    }
  } else {path_pair <- list()}

  if (length(pos_nopath_pair) >= 1){
  # Check for inconsistencies in pairs without path
    nopath_pair <- list()
    for (i in c(1:length(pos_nopath_pair))){
      tIndex <- pos_nopath_pair[[i]]
      if (tIndex %% ncol(pk) == 0){
        tRow <- ncol(pk)
        tCol <- tIndex %/% ncol(pk)
      } else {
        tRow <- tIndex %% ncol(pk)
        tCol <- tIndex %/% ncol(pk) + 1
      }
      nopath_pair[[i]] <- sort(c(tCol, tRow))
    }
    nopath_pair <- unique(nopath_pair)
  } else {nopath_pair <- list()}
```

```
  all_pairs <- c(path_pair, nopath_pair)

  if (length(all_pairs)==0) {
    # if no pairs are extracted from the specified prior knowledge
    # discard the prior knowledge.
    self$pk <- NULL
    return(NULL)
  }
  out <- as.data.frame(matrix(c(unlist(all_pairs)),ncol=2,byrow=T))
  colnames(out) <- c("From","To")
  return(out)
},


discover_causal_order = function(X){

  # Specify the number of latent variables
  q <- dim(self$lambda)[2]


  # Apply bonferroni correction
  thresh_p <- self$alpha / (q-1)


  # Search causal orders one by one from the bottom upward
  return(self$search_causal_order(X, thresh_p))
},


residual = function(xi, xj){
  # The residual when xi is regressed on xj.
  xi - (cov(xi, xj) / var(xj)) * xj
},


search_candidate = function(U){
  # Search for candidate features
  # If no prior knowledge is specified, nothing to do.
  if (is.null(self$pk)){return(U)}

  # Candidate features that are not to the left of the partial order
  diff_U <- unlist(self$partial_orders[,1])
  Uc <- setdiff(U, diff_U)
  return(Uc)
},


search_causal_order = function(X, thresh_p){

  # Search causal orders one by one from the bottom upward.

  # Number latent variables
  U <- c(1:ncol(X))

  # Substantiate empty lists of K_bttm and p_bttm
  K_bttm <- c()
  p_bttm <- c()


  is_search_causal_order <- TRUE
```

```
      # If only one variable is left in list U, enter the top of K_bttm
      if(length(U) <= 1){
        K_bttm <- c(U, K_bttm)
        p_bttm <- c(0, p_bttm)
        is_search_causal_order <- FALSE}


      while (is_search_causal_order) {

        # Search for candidate features
        Uc <- self$search_candidate(U)


        if (length(Uc) == 1){

          # If there is only one variable in Uc,
          # calculate HSIC with the rest of the variables
          m <- c(Uc[1])
          predictors <- setdiff(U, Uc[1])
          res <- self$compute_residuals(X, predictors, m)
          fisher_p <- self$fisher_hsic_test(X[,predictors],res,Inf)[1]

        } else {

          # Find the most sink variable
          exo_out <- self$find_exo_vec(X, Uc)

          # Index of the sink variable = m
          m <- as.numeric(exo_out["m"])

          # p-value of HSIC with Fisher's method = fisher_p
          fisher_p <- as.numeric(exo_out["max_p"])
        }

        # Conduct statistical test by the p-value or the statistic
        # If statistical test is not rejected
        if (fisher_p >= thresh_p) {


          # Add index of the sink variable to the top of K_bttm
          K_bttm <- c(m, K_bttm)


          # Add fisher's p-value of the sink variable to the top of p_bttm
          p_bttm <- c(fisher_p, p_bttm)


          # Update U by dropping the index of the sink variable (m)
          U <- U[U != m]


          # Update the partial order
          if (is.null(self$pk) == FALSE){
            self$partial_orders <-
self$partial_orders[which(self$partial_orders[,1]!=m),]
          }


          # If there is only one candidate left, end the search
          if (length(U) <= 1){
            K_bttm <- c(U, K_bttm)
```

```
            p_bttm <- c(0.0, p_bttm)
            is_search_causal_order <- FALSE}

        } else {

          # If statistical test is rejected
          is_search_causal_order <- FALSE
        }
      }

      out <- c()
      out$K_bttm <- K_bttm
      out$p_bttm <- p_bttm
      return(out)
    },


    find_exo_vec = function(X, U){
      # Find the most exogenous vector.
      max_p <- -Inf
      max_p_stat <- Inf

      exo_vec <- c()
      for (j in (1:length(U))){
        xi_index <- setdiff(U,U[j])
        xj_index <- c(U[j])


        # Compute residuals
        res <- self$compute_residuals(X, xi_index, xj_index)


        # HSIC test with Fisher's method to test the independence between
        # a set of predictor (xi_index) and error term (res)
        fisher_hsic_out <-
self$fisher_hsic_test(X[,xi_index],res,max_p_stat)


        # Retrieve the p-value of HSIC with Fisher's method = fisher_p
        fisher_p <- fisher_hsic_out[1]


        # Retrieve the statistic of HSIC with Fisher's method = fisher_stat
        fisher_stat <- fisher_hsic_out[2]


        # If the fisher_p of j is the greatest so far, update
        # the index of explanatory variables (exo_vec),
        # the maximum of fisher_p (max_p), and;
        # the maximum of fisher_stat (max_p_stat)
        if (fisher_stat < max_p_stat | fisher_p > max_p){
          exo_vec <- xi_index
          max_p <- fisher_p
          max_p_stat <- fisher_stat
        }
      }

      # Obtain the index of sink variable that is not included in exo_vec =
m
      m <- setdiff(U, exo_vec)
      out <- c(m, exo_vec, max_p) %>%
```

```
        'names<-'(c("m","exo_vec","max_p"))
      return(out)
    },


    compute_residuals = function(X, predictors, target){
      # Compute residuals
      if (is.null(self$reg)){
        # Compute residuals of least square regressions
        cov <- cov((X))
        term1 <- MASS::ginv(cov[predictors,predictors])
        term2 <- matrix(
          cov[target, predictors],
          nrow = length(predictors),
          ncol = 1)
        coef <- term1 %*% term2
        res <- as.matrix(X[,target]) - as.matrix(X[,predictors]) %*% coef
      } else {
        lm_fit <- lm(X[,target] ~., data = X[,predictors])
        res <- lm_fit$residuals
      }

      return(res)
    },


    fisher_hsic_test = function(X, res, max_p_stat){
      # Conduct statistical test by HSIC with Fisher's methods.

      # Substantiate the statistic of HSIC with Fisher's method =
fisher_stat
      fisher_stat <- 0


      if (is.null(ncol(X))){
        # If X is a vector, set q = 1
        q <- 1
      } else {
        # If X is a matrix, set q = its columm length
        q <- ncol(X)}

      if (q == 1){
        # If there is only one predictor, simply HSIC will do
        hsic_out <- dHSIC::dhsic.test(
          X = X,
          Y = res,
          kernel = "gaussian",
          method = "gamma")

        # Retrieve p-value of HSIC (without Fisher's method) = fisher_p
        fisher_p <- hsic_out$p.value

        # Retrieve the statistic of HSIC (without Fisher's method) =
fisher_stat
        fisher_stat <- hsic_out$statistic

      } else {

        # If there is more than one predictor, turn HSIC with Fisher's
method
        for (i in c(1:q)){
```

```
          # Conduct pairwise HSIC and retrieve the p-value of HSIC
          hsic_p <- dHSIC::dhsic.test(
            X = as.data.frame(X[,i]),
            Y = res,
            kernel = "gaussian",
            method = "gamma")$p.value


          if (hsic_p == 0){
            # If p-value of HSIC is zero, set fisher_stat as infinite
            fisher_stat <- Inf
          } else {

            # If p-value of HSIC is non-zero, add to fisher_stat
            fisher_stat <- fisher_stat + (-2 * log(hsic_p))
          }

          if (fisher_stat > max_p_stat){break}
        }

        # Compute the p-value of HSIC with Fisher's method by doing chi-
square test
        fisher_p <- pchisq(fisher_stat, df = (2 * q), lower.tail=FALSE)
      }
      return(c(fisher_p, fisher_stat) %>% 'names<-
'(c("fisher_p","fisher_stat")))
    },


    estimate_structural_matrix = function(X){

      if (is.null(self$pk) == FALSE){
        tmp_pk <- self$pk
        diag(tmp_pk) <- 0
      }

      q <- dim(self$lambda)[2]

      B <- matrix(0, nrow = q, ncol = q)

      for (i in 2:length(self$causal_order)){
        target <- unlist(self$causal_order[i])
        predictors <- unlist(self$causal_order[1:(i-1)])

        # Exclude variables specified in no_path with prior knowledge
        if(!is.null(self$pk)){
          predictors_pk <- c()
          for (p in predictors){
            if (tmp_pk[target, p] != 0){predictors_pk <- c(predictors_pk,
p)}
          }
          predictors <- predictors_pk
        }

        if(length(predictors) > 0){
          B[target, predictors] <- self$my_regsem(X,predictors,target)
        }
      }
      return(B)
    },
```

```
    my_regsem = function(X, predictors, target){


      # Prepare syntax for measurement model (MM)
      syntax_MM <- paste(self$model_pure[c(predictors,target)],collapse =
"\n")


      # Prepare syntax for structural model (SM)
      list_xj <- unlist(self$listLV[c(predictors)])
      xi <- unlist(self$listLV[c(target)])

      syntax_SM <- ""; count <- 0
      for (xj in list_xj){
        count <- count + 1
        tP <- paste(xi," ~ c",count,"*",xj,";\n",sep="")
        syntax_SM <- paste(syntax_SM,tP,sep="")
      }

      # Combine MM with SM
      syntax_SEM <- paste(syntax_MM,syntax_SM,sep="\n")


      # Subset dataset
      X_ <- X %>%
        subset(select = c(unlist(self$listOV[c(predictors,target)])))


      # Run a standard SEM
      sem_out <- lavaan::sem(syntax_SEM,X_,fixed.x = FALSE)


      # If there is more than one predictor, run a regularized SEM
      if (length(predictors) >=2) {
        regsem_out <- regsem::cv_regsem(
          sem_out,
          n.lambda = self$regsem_nlambda,
          type = self$regsem_penalty,
          jump = self$regsem_jump,
          max.iter = 100,
          step = 0.3,
          pars_pen = c(paste("c",c(1:count),sep="")),
          verbose = FALSE)


        # Extract all parameter estimates (pe) after penalizations
        pe <- regsem_out$final_pars


        # Select parameter estimates
        pe_selected <- c()
        for (i in c(1:length(predictors))){
          pe_selected[i] <- pe[paste(list_xj[i], " -> ", xi,sep="")]
        }

      } else {

        # If there is only one predictor, no need to run RegSEM but simply
obtain pe from SEM
```

```
        pe <- lavaan::parameterEstimates(sem_out)
        tmp_row <- which(pe$lhs == xi & pe$op == "~" & pe$rhs ==
list_xj[1])
        pe_selected <- pe[tmp_row,"est"]
      }

      return(matrix(pe_selected, ncol = length(predictors)))
    },


    my_regsem_final = function(X, B){


      # Run an overall RegSEM
      listLV <- self$listLV
      B_ <- B
      B_[B_ > 0.00] <- 1.00
      num_para <- length(B_[B_ == 1])


      # Draft syntax for structural model (SM)
      syntax_SM <- ""; count <- 0;
      for (i in c(1:length(listLV))){
        for (j in c(setdiff(c(1:length(listLV)),i))){
          if(B_[i,j] == 1){
            count <- count + 1
            ts <- paste(listLV[[i]], " ~
c",count,"*",listLV[[j]],";",sep="")
            syntax_SM <- paste(syntax_SM, ts, sep = "\n")
          }
        }
      }


      # Draft syntax for measurement model (MM)
      if (!is.null(self$model_other)){
        syntax_MM <- paste(
          stringr::str_flatten(self$model_pure, collapse = "\n"),
          self$model_other, collapse = "\n")
      } else {
        syntax_MM <- stringr::str_flatten(self$model_pure, collapse = "\n")
      }


      # Combine model syntax
      syntax_SEM <- paste(syntax_SM, syntax_MM, sep = "\n")


      # Run a standard SEM
      sem_out <- lavaan::sem(syntax_SEM, X)


      # Run a regularized SEM
      regsem_out <- regsem::cv_regsem(
        sem_out,
        n.lambda = self$regsem_nlambda,
        type = self$regsem_penalty,
        jump = self$regsem_jump,
        max.iter = 100,
        step = 0.3,
        pars_pen = c(paste("c",c(1:num_para),sep="")),
```

```
        verbose = FALSE)


      # Retrieve all parameter estimates (pe)
      pe <- regsem_out$final_pars[1:num_para]


      # Substantiate a null matrix
      beta <- matrix(0, nrow = length(listLV), ncol = length(listLV))


      # Fill in the null matrix
      for (i in c(1:length(listLV))){
        for (j in c(setdiff(c(1:length(listLV)),i))){
          if(B_[i,j] == 1){
            beta[i,j] <- pe[[paste(listLV[[j]], " -> ",
listLV[[i]],sep="")]]
          }
        }
      }

      return(beta)

    }
  )
)

#----Latent-Variable RCD ----

#' @title RCD_LV class
#' @description R implementation of repetitive causal discovery for latent
variables
#' @references T.N.Maeda and S.Shimizu. RCD: Repetitive causal discovery of
linear non-Gaussian acyclic models with latent confounders. In Proc. 23rd
International Conference on Artificial Intelligence and Statistics
(AISTATS2020), Palermo, Sicily, Italy. PMLR  108:735-745, 2020.
#' @export

RCD_LV <- R6::R6Class(
  "RCD_LV",
  public = list(

    #' @param regsem_penalty (string) Penalty type for regsem::cv_regsem
function (Default: lasso)
    #' @param regsem_n.lambda (numeric) Number of penalization values to
test for regsem::cvregsem function (Default: 20)
    #' @param regsem_jump (numeric) Amount to increase penalization each
iteration for regsem::cvregsem function (Default: .03)
    #' @param mat_lambda (matrix) A matrix of relations between item and
factors, with dimension of items (p) by factors (q)
    #' @param mat_phi (matrix) A squared matrix of relations between
factors, with dimension of factors (q) by factors (q)
    #' @param syntax_pmm (vector of strings) Syntax for Pure Measurement
Model, with dimension of factors (q) by 1
    #' @param syntax_snmm (scalar of string, optional) Syntax for
Structural Null Measurement Model
    #' @param pk (matrix) A squared matrix of prior knowledge, with
dimension of factors (q) by factors (q)
    #' @param list_method_fs (vector of string) Approaches to approximating
factor scores (Default: Thurstone, tenBerge, Bartlett)
```

```
lambda = NULL,
phi = NULL,
model_pure = NULL,
model_other = NULL,
regsem_penalty = "alasso",
regsem_nlambda = 20,
regsem_jump = 0.03,
pk = NULL,
fsa = c("Thurstone","tenBerge","Bartlett"),
fsa_users = "tenBerge",

max_explanatory_num = 2,
alpha_cor = 0.01,
alpha_ind = 1,
alpha_shapiro = 0.01,
indep_test2 = FALSE,
k_iter = c(0.5,0.2,0.1,0.01),

adjacency_matrix = NULL,
partial_orders = NULL,
list_ancestor = NULL,
list_ancestors_fsa = list(),
list_parents = NULL,
list_parents_fsa = list(),
list_confounders = NULL,
list_confounders_fsa = list(),
history_alpha_ind = NULL,
history_alpha_ind_fsa = list(),
listLV = NULL,
listOV = NULL,


initialize = function(
  lambda = NULL,
  phi = NULL,
  model_pure = NULL,
  model_other = NULL,
  regsem_penalty = "alasso",
  regsem_nlambda = 20,
  regsem_jump = 0.03,
  pk = NULL,
  fsa = c("Thurstone","tenBerge","Bartlett"),
  fsa_users = "tenBerge",
  max_explanatory_num = 2,
  alpha_cor = 0.01,
  alpha_ind = 1,
  alpha_shapiro = 0.01,
  indep_test2 = FALSE,
  k_iter = c(0.5,0.2,0.1,0.01)){

  self$lambda <- lambda
  self$phi <- phi
  self$model_pure <- model_pure
  self$model_other <- model_other
  self$regsem_penalty <- regsem_penalty
  self$regsem_nlambda <- regsem_nlambda
  self$regsem_jump <- regsem_jump
  self$pk <- pk
  self$fsa <- fsa
  self$fsa_user <- fsa_user
  self$max_explanatory_num <- max_explanatory_num
```

```
  self$alpha_cor <- alpha_cor
  self$alpha_ind <- alpha_ind
  self$alpha_shapiro <- alpha_shapiro
  self$indep_test2 <- indep_test2
  self$k_iter <- k_iter
},

fit = function(X){

  # Specify the number of latent variables
  q <- dim(self$lambda)[2]

  # Issue warnings
  if(self$max_explanatory_num <= 0){
    warning("Maximum number of explanatory variable must be >0.")}
  if(self$alpha_cor < 0){warning("alpha_cor must be >=0.")}
  if(self$alpha_ind <0){warning("alpha_ind must be >=0.")}
  if(self$alpha_shapiro <0){warning("alpha_shapiro must be >=0.")}


  # Check prior knowledge
  if(is.null(self$pk) == FALSE){
    if(dim(self$pk)[1] != q |
       dim(self$pk)[2] != q){
      sprintf("The shape of prior knowledge must be (%d, %d)", q, q)
    }
  }


  # Name latent variables and observed variables
  self$name_variables()


  # Check if the user-specified fsa is on the list of fsa
  if (!(self$fsa_user %in% self$fsa)){

    # If not, renew the list
    self$fsa <- c(self$fsa, self$fsa_user)
  }


  for (i_fsa in self$fsa){


    # Approximate factor scores
    X_fsa <- X %>% self$compute_factor_scores(i_fsa)


    # Discover ancestors, parents & confounders
    apc_out <- X_fsa %>% self$discover_apc()


    # Log ancestors list
    self$list_ancestors_fsa[[i_fsa]] <- apc_out$M


    # Log parents list
    self$list_parents_fsa[[i_fsa]] <- apc_out$P


    # Log confounders list
```

```
        self$list_confounders_fsa[[i_fsa]] <- apc_out$C


        # Log alpha_ind_history
        self$history_alpha_ind_fsa[[i_fsa]] <- apc_out$C_num_pairs_history

      }

      # Check if all lists are consistent across FSA
      is_consistent <- TRUE
      for (i_fsa in self$fsa){
        for (j_fs in setdiff(self$fsa, i_fsa)){

          A_i_fsa <- self$list_ancestors_fsa[[i_fsa]]
          A_j_fsa <- self$list_ancestors_fsa[[j_fsa]]
          P_i_fsa <- self$list_parents_fsa[[i_fsa]]
          P_j_fsa <- self$list_parents_fsa[[j_fsa]]
          C_i_fsa <- self$list_confounders_fsa[[i_fsa]]
          C_j_fsa <- self$list_confounders_fsa[[j_fsa]]

          if ((!identical(A_i_fsa, A_j_fsa)) |
              (!identical(P_i_fsa, P_j_fsa)) |
              (!identical(C_i_fsa, C_j_fsa)) ){
            is_consistent <- FALSE; break
          }
        }
      }

      # If all lists are identical, endorse any
      if (is_consistent){
        self$list_ancestors <- self$list_ancestors_fsa[[1]]
        self$list_parents <- self$list_parents_fsa[[1]]
        self$list_confounders <- self$list_confounders_fsa[[1]]
        self$history_alpha_ind <- self$history_alpha_ind_fsa[[1]]
      } else {

        # If not identical, endorse the user-specified approach
        self$list_ancestors <- self$list_ancestors_fsa[[self$fsa_user]]
        self$list_parents <- self$list_parents_fsa[[self$fsa_user]]
        self$list_confounders <- self$list_confounders_fsa[[self$fsa_user]]
        self$ind_alpha_history <-
self$history_alpha_ind_fsa[[self$fsa_user]]
      }


      # Estimate a preliminary structural matrix
      B_preliminary <- self$estimate_structural_matrix(X,
self$list_parents, self$list_confounders)


      # Finalize structural matrix
      self$adjacency_matrix <- self$my_regsem_final(X, B_preliminary)
    },

    name_variables = function(){


      # Specify the number of latent variables
      q <- dim(self$lambda)[2]
```

```
      # Identify latent variables and their corresponding observed
variables
      listLV <- list(); listOV <- list()


      for (i_LV in c(1:q)){

        # Find out the latent variable label
        tmp_LV <- gsub("=~.*","",self$model_pure[i_LV])


        # Remove spacing from the label
        self$listLV[[i_LV]] <- gsub(" ", "", tmp_LV, fixed = TRUE) %>%
          as.character()


        # Find out the corresponding observed variables
        tmp_OVs <- gsub(".*=~","",self$model_pure[i_LV])


        # And convert it into a vector
        tmp_OVs <- gsub(" ", "", tmp_OVs, fixed = TRUE)
        self$listOV[[i_LV]] <- unlist(strsplit(tmp_OVs,"\\+"))

      }
    },


    compute_factor_scores = function(X, method){

return(psych::factor.scores(X,self$lambda,self$phi,method=method,impute="me
dian")$scores)
    },


    extract_partial_orders = function(pk){

      # Extract partial orders from prior knowledge.
      pos_path_pair <- which(pk == 1)
      pos_nopath_pair <- which(pk == 0)

      # Check for inconsistencies in pairs with path
      check_path_pair <- c()
      for (i in c(1:length(pos_path_pair))){
        tIndex <- pos_path_pair[i]
        if (tIndex %% ncol(pk) == 0){
          tRow <- ncol(pk)
          tCol <- tIndex %/% ncol(pk)
        } else {
          tRow <- tIndex %% ncol(pk)
          tCol <- tIndex %/% ncol(pk) + 1
        }
        check_path_pair <- rbind(check_path_pair,c(tCol, tRow))
        check_path_pair <- rbind(check_path_pair,c(tRow, tCol))
      }

      if(sum(duplicated(check_path_pair) >= 1)){
        cat("The prior knowledge contains inconsistencies (Col, Row):\n")
        message(check_path_pair[duplicated(check_path_pair),])
      }
```

```
      path_pair <- check_path_pair[seq(1,nrow(check_path_pair),2),]

      # Check for inconsistencies in pairs without path
      # If there are duplicate pairs without path, they cancel out and are
not ordered.
      check_nopath_pair <- list()
      for (i in c(1:length(pos_nopath_pair))){
        tIndex <- pos_nopath_pair[i]
        if (tIndex %% ncol(pk) == 0){
          tRow <- ncol(pk)
          tCol <- tIndex %/% ncol(pk)
        } else {
          tRow <- tIndex %% ncol(pk)
          tCol <- tIndex %/% ncol(pk) + 1
        }
        check_nopath_pair <- rbind(check_path_pair,c(tCol, tRow))
        check_nopath_pair <- rbind(check_path_pair,c(tRow, tCol))
      }

      nopath_pair <- unique(check_nopath_pair)

      all_pairs <- rbind(path_pair, nopath_pair)

      if (nrow(all_pairs) == 0) {
        # if no pairs are extracted from the specified prior knowledge
        # discard the prior knowledge.
        self$pk = NULL
        return(NULL)
      }

      all_pairs <- unique(all_pairs)
      return(all_pairs)
    },


    discover_apc = function(X){

      C_num_pairs_history <- c()

      for (k_alpha in self$k_iter){

        self$ind_alpha <- k_alpha

        # Determine ancestors
        M <- self$extract_ancestors(X)


        # Determine parents
        P <- self$extract_parents(X, M)


        # Determin confounders
        C <- self$extract_vars_sharing_confounders(X, P)
        C_num_pairs_history <- c(C_num_pairs_history,
sum(lengths(C[!is.na(C)])))
      }

      out <- c()

      # Here is the finalized independence of alpha
      out$ind_alpha <- self$k_iter[which.min(C_num_pairs_history)]
```

```
      out$C_num_pairs_history <- C_num_pairs_history


      # Extract a set of ancestors of each variable
      out$M <- self$extract_ancestors(X)

      # Extract parents (direct causes) from the set of ancestors.
      out$P <- self$extract_parents(X, M)

      # Find the pairs of variables affected by the same latent confounders
      out$C <- self$extract_vars_sharing_confounders(X, P)

      return(out)
    },

    get_common_ancestors = function(M, U){
      # Get the set of common ancestors of U
      Mj_list <- list(); c = 0
      for (xj in U){
        c = c + 1; Mj_list[[c]] <- M[[xj]]
      }

      out <- Reduce(intersect, Mj_list)
      if(length(out) == 0){return(NA)} else {
        return(Reduce(intersect, Mj_list))
      }
    },


    get_resid_and_coef = function(X, endog_idx, exog_idcs){
      # Get the residuals and coefficients of the ordinary least square
method
      X <- as.data.frame(X)
      fml <- as.formula(paste0(names(X)[endog_idx], "~."))
      X_ <- X[names(X)[c(endog_idx, exog_idcs)]]
      lr <- lm(fml, data = X_)
      out <- list()
      out$resid <- lr$residuals
      out$coef <- lr$coefficients
      return(out)
    },


    get_residual_matrix = function(X, U, H_U){
      if (length(H_U) == 1 & all(is.na(H_U)) == TRUE){return(X)}
      if (all(is.na(H_U))){return(X)}

      Y <- X
      Y[,] <- 0
      for (xj in U){
        Y[,xj] <- self$get_resid_and_coef(X,xj,H_U)$resid
      }
      return(Y)
    },


    is_non_gaussianity = function(Y, U){
      # Test whether a variable is generated from a non-Gaussian process
using the Shapiro-Wilk test
      for (xj in U){
```

```
        if (shapiro.test(Y[,xj])$p.value >
self$shapiro_alpha){return(FALSE)}
      }
      return(TRUE)
    },

    is_correlated = function(a, b){
      # Estimate that the two variables are linearly correlated
      return(Hmisc::rcorr(a,b,type = c("pearson"))$P[1,2] < self$cor_alpha)
    },

    exists_ancestor_in_U = function(M,U,xi,xj_list){
      # Check if xi is not in Mj, the ancestor of xj.
      for (xj in xj_list){
        if (xi %in% M[[xj]]){return(TRUE)}
      }

      # Check if xj_list is a subset of Mi, the ancestor of xi.
      if (all(is.na(M[[xi]])) == FALSE){
        if (all(xj_list == intersect(xj_list, M[[xi]]))){return(TRUE)}
      }
      return(FALSE)
    },

    is_independent = function(X,Y){
      fisher_p <- dHSIC::dhsic.test(
        X, Y, method = "gamma", kernel = "gaussian")$p.value
      return(fisher_p > self$ind_alpha)
    },

    is_independent_of_resid = function(Y, xi, xj_list){
      # Check whether the residuals obtained from multiple regression
      n_samples <- nrow(Y)

      # Multiple Regression with OLS.
      is_all_independent <- TRUE
      resid <- self$get_resid_and_coef(Y,xi,xj_list)$resid
      for (xj in xj_list){
        if (self$is_independent(resid, Y[,xj]) == FALSE){
          is_all_independent <- FALSE
          break
        }
      }

      return(is_all_independent)
    },

    is_independent_of_resid2 = function(Y, xi, xj_list){
      U <- c(xi, xj_list); n_features <- length(U)
      K <- c()

      for (a in c(1:n_features)){
        m <- self$search_causal_order(Y,U)
        for (i in U){
          if (i != m){
            Y[,i] <- self$residual(Y[,i],Y[,m])
          }
        }
        K <- c(K, m)
        U <- U[U != m]
      }
```

```
      return(K[n_features]== xi)
    },

    extract_ancestors = function(X){
      # Extract a set of ancestors of each variable
      n_features <- ncol(X)
      M <- list()
      for (i in c(1:n_features)){M[[i]] <- c(NA)}
      l <- 1
      hu_history <- list()

      while(TRUE){
        changed <- FALSE
        U_list <- combn(c(1:n_features),l+1,simplify = FALSE)
        for (U in U_list){
          U <- sort(U)

          # Get the set of common ancestors of U
          H_U <- self$get_common_ancestors(M, U)

          str_U <- as.character(paste(U,collapse = " "))

          if (is.null(hu_history[[str_U]]) == FALSE){
            if (all(is.na(H_U))==FALSE &
all(is.na(hu_history[[str_U]]))==FALSE){
              if (identical(H_U, hu_history[[str_U]])) {next}
            }
          }

          Y <- as.data.frame(self$get_residual_matrix(X, U, H_U))

          # Test whether a variable is generated from a non-Gaussian
process using the Shapiro-Wilk test
          if (self$is_non_gaussianity(Y,U) == FALSE){next}

          # Estimate that the two variables are linearly correlated using
the Pearson's Correlation
          is_cor <- TRUE
          for (i in combn(U,2, simplify = FALSE)){
            xi <- i[1]; xj <- i[2]
            if (self$is_correlated(Y[,xi], Y[,xj]) == FALSE){
              is_cor <- FALSE
              break
            }
          }
          if (is_cor == FALSE){next}

          sink_set <- c()
          for (xi in U){
            xj_list <- setdiff(U,xi)
            if(self$exists_ancestor_in_U(M,U,xi,xj_list)){next}

            # Check whether the residuals obtained from multiple
regressions are independent
            if(self$indep_test2 == FALSE){
              if(self$is_independent_of_resid(Y,xi,xj_list)){
                sink_set <- c(sink_set, xi)
              }
            } else {
              if(self$is_independent_of_resid2(Y,xi,xj_list)){
```

```
            sink_set <- c(sink_set, xi)
          }
        }
      }

      if (length(sink_set)==1){
        xi <- sink_set[1]
        xj_list <- setdiff(U,xi)

        if (all(is.na(M[[xi]]))){
          M[[xi]] <- xj_list
          changed <- TRUE
        }

        if (all(is.na(M[[xi]])) == FALSE){
          if(identical(M[[xi]], union(M[[xi]], xj_list)) == FALSE){
            M[[xi]] <- union(M[[xi]], xj_list)
            changed <- TRUE
          }
        }
      }

      hu_history[[str_U]] <- H_U
    }

    if (changed){
      l <- 1
    } else if (l < self$max_explanatory_num){
      l  <- l + 1
    } else {
      break
    }
  }
  return(M)
},

is_parent = function(X, M, xj, xi){
  if (length(setdiff(M[[xi]], xj)) > 0){
    zi <- self$get_resid_and_coef(X,xi,setdiff(M[[xi]],xj))$resid
  } else {
    zi <- X[,xi]
  }

  if (length(intersect(M[[xi]], M[[xj]])) > 0){
    wj <-
self$get_resid_and_coef(X,xj,intersect(M[[xi]],M[[xj]]))$resid
  } else {
    wj <- X[,xj]
  }

  # Check if zi and wj are correlated
  return(self$is_correlated(wj, zi))
},

extract_parents = function(X, M){
  # Extract parents (direct causes) from a set of ancestors.
  n_features <- ncol(X)
  P <- list()
  for (i in c(1:n_features)){P[[i]] <- c(NA)}

  for (xi in c(1:n_features)){
```

```
    if (all(is.na(M[[xi]]))==FALSE){
      for (xj in M[[xi]]){
        # Check if xj is the parent of xi.
        if (self$is_parent(X, M, xj, xi) == TRUE){
          P[[xi]] <- unique(c(P[[xi]], xj))
          P[[xi]] <- P[[xi]][!is.na(P[[xi]])]
        }
      }
    }
  }
  return(P)
},

get_resid_to_parent = function(X,idx,P){
  if (all(is.na(P[[idx]]))){return(X[,idx])}
  return(self$get_resid_and_coef(X,idx,c(P[[idx]]))$resid)
},

extract_vars_sharing_confounders = function(X,P){
  # Find the pairs of variables affect by the same latent confounders.
  n_features <- ncol(X)
  C <- list()
  for (i in c(1:n_features)){C[[i]] <- c(NA)}

  for (k in combn(c(1:n_features),2, simplify = FALSE)){
    i <- k[1]; j <- k[2]
    if (i %in% P[[j]] | j %in% P[[i]]){
      next
    }
    resid_xi <- self$get_resid_to_parent(X, i, P)
    resid_xj <- self$get_resid_to_parent(X, j, P)
    if (self$is_correlated(resid_xi, resid_xj)){
      C[[i]] <- unique(c(C[[i]],j))
      C[[j]] <- unique(c(C[[j]],i))
      C[[i]] <- C[[i]][!is.na(C[[i]])]
      C[[j]] <- C[[j]][!is.na(C[[j]])]
    }
  }
  return(C)
},

estimate_coarse_adjacency_matrix = function(X, P, C){

  # Check parents
  n_features <- length(self$syntax_pmm)

  if(is.null(self$pk) == FALSE){
    tmp_pk <- self$pk
    diag(tmp_pk) <- 0
  }

  B <- matrix(0, nrow = n_features, ncol = n_features)
  for (xi in c(1:n_features)){
    xj_list <- c(P[[xi]])
    if (all(is.na(xj_list))){next}

    xj_list <- sort(xj_list)

    # Exclude variables specified in no_path with prior knowledge
    if(is.null(self$pk) == FALSE){
      xj_list_pk <- c()
```

```
      for (p in xj_list){
        if (tmp_pk[xi, p] != 0){xj_list_pk = c(xj_list_pk, p)}
      }
      xj_list <- xj_list_pk
    }

    coef <- self$predictRegSEM(X, xj_list, xi)

    k <- 0
    for (xj in xj_list){
      k <- k + 1
      B[xi, xj] <- coef[,k]
    }
  }

  # Check confounders
  for (xi in c(1:n_features)){
    xj_list <- sort(c(C[[xi]]))
    if (all(is.na(xj_list))){next}

    for (xj in xj_list){
      B[xi, xj] <- NA
    }
  }

  return(B)
},

predictRegSEM = function(X, predictors, target){

  # Prepare Syntax
  tmp_Measurement <- paste(
    self$syntax_pmm[c(predictors,target)],collapse = "\n")

  xj_list <- unlist(self$latent_variables[c(predictors)])
  xi <- unlist(self$latent_variables[c(target)])

  tmp_Path <- ""; count <- 0
  for (xj in xj_list){
    count <- count + 1
    tP <- paste(xi," ~ c",count,"*",xj,";\n",sep="")
    tmp_Path <- paste(tmp_Path,tP,sep="")
  }
  tmp_syntax <- paste(tmp_Measurement,tmp_Path,sep="\n")

  # Prepare Dataset
  tmp_dataset <- subset(
    X, select = c(unlist(self$item_indicators[c(predictors,target)])))

  # Run Standard SEM
  tmp_lav_out <- lavaan::sem(tmp_syntax,tmp_dataset,fixed.x = FALSE)

  if (length(predictors) >=2) {
    tmp_reg_out <- regsem::cv_regsem(
      tmp_lav_out,
      n.lambda = self$regsem_n.lambda,
      type = self$regsem_penalty,
      jump = self$regsem_jump,
      max.iter = 100,
      step = 0.3,
      pars_pen = c(paste("c",c(1:count),sep="")),
```

```
          verbose = FALSE)

        tmp_coef_all <- tmp_reg_out$final_pars

        tmp_coef <- c()
        for (i in c(1:length(predictors))){
          tmp_coef[i] <- tmp_coef_all[paste(xj_list[i], " -> ", xi,sep="")]
        }
      } else {
        tmp_coef_all <- lavaan::parameterEstimates(tmp_lav_out)
        tmp_row <- which(
          tmp_coef_all$lhs == xi &
            tmp_coef_all$op == "~" &
            tmp_coef_all$rhs == xj_list[1])
        tmp_coef <- tmp_coef_all[tmp_row,"est"]
      }

      tmp_coef <- matrix(tmp_coef, ncol = length(predictors))
      return(tmp_coef)
    },

    estimate_refined_adjacency_matrix = function(X,
coarse_adjacency_matrix){

      ## Objective: Run an overall regularized SEM
      lat_var <- self$latent_variables
      adj_mat_unit <- coarse_adjacency_matrix
      adj_mat_unit[adj_mat_unit > 0.00] <- 1.00
      adj_mat_unit[is.na(adj_mat_unit)] <- .00
      num_parameter <- length(adj_mat_unit[adj_mat_unit == 1])

      ## Draft the structural model
      count <- 0; tmp_SM <- ""
      for (i in c(1:length(lat_var))){
        for (j in c(setdiff(c(1:length(lat_var)),i))){
          if(adj_mat_unit[i,j] == 1){
            count <- count + 1
            ts <- paste(lat_var[[i]], " ~
c",count,"*",lat_var[[j]],";",sep="")
            tmp_SM <- paste(tmp_SM, ts, sep = "\n")
          }
        }
      }

      ## Draft the measurement model
      if(is.null(self$syntax_snmm)){
        tmp_MM <- paste(self$syntax_pmm, collapse = "\n")
      } else {
        tmp_MM <- self$syntax_snmm
      }

      tmp_syntax <- paste(tmp_SM, tmp_MM, sep = "\n")
      lav_out <- lavaan::sem(tmp_syntax, X)
      reg_out <- regsem::cv_regsem(
        lav_out,
        n.lambda = self$regsem_n.lambda,
        type = self$regsem_penalty,
        jump = self$regsem_jump,
        max.iter = 100,
        step = 0.3,
        pars_pen = c(paste("c",c(1:num_parameter),sep="")),
```

```
        verbose = FALSE)

      coef_all <- reg_out$final_pars[1:num_parameter]
      beta_mat <- matrix(0.00, nrow = length(lat_var), ncol =
length(lat_var))

      for (i in c(1:length(lat_var))){
        for (j in c(setdiff(c(1:length(lat_var)),i))){
          if(adj_mat_unit[i,j] == 1){
            tmp_head <- paste(lat_var[[j]], " -> ", lat_var[[i]],sep="")
            beta_mat[i,j] <- coef_all[[tmp_head]]
          }
        }
      }

      return(beta_mat)

    },

    residual = function(xi,xj){
      # The residual when xi is regressed on xj.
      xi - (cov(xi, xj) / var(xj)) * xj
    },

    entropy = function(u){
      # Calculate entropy using the maximum entropy approximation
      k1 <- 79.047
      k2 <- 7.4129
      gamma <- 0.37457
      return(
        (1 + log(2 * pi)) / 2 -
          k1 * (mean(log(cosh(u))) - gamma)^2 -
          k2 * (mean(u * exp((-u^2) / 2)))^2)
    },

    diff_mutual_info = function(xi_std, xj_std, ri_j, rj_i){
      # Calculate the difference of the mutual informations.
      term1 <- self$entropy(xj_std) + self$entropy(ri_j / sd(ri_j))
      term2 <- self$entropy(xi_std) + self$entropy(rj_i / sd(rj_i))
      return(term1 - term2)
    },

    search_causal_order = function(X, U){
      M_list <- c()
      for (i in U){
        M = 0
        for (j in U){
          if (i != j){
            xi_std = (X[, i] - mean(X[, i])) / sd(X[, i])
            xj_std = (X[, j] - mean(X[, j])) / sd(X[, j])
            ri_j <- self$residual(xi_std, xj_std)
            rj_i <- self$residual(xj_std, xi_std)

            M = M + min(0,self$diff_mutual_info(xi_std, xj_std, ri_j,
rj_i))^2
          }
        }
        M_list <- c(M_list, -M)
      }
      return(U[which.max(M_list)])
```

```
    }
  )
)


#---- ICA-LiNGAM ----------


ICALiNGAM <- R6::R6Class(
  "ICALiNGAM",
  public = list(

    #' @title ICALiNGAM class
    #' @description
    #' To discover the causal relations between observed factors using ICA-
LiNGAM algorithm for observed variables
    #' @param max_k a single numeric that specifies the maximum attempt to
search causal order (Default: 3)
    #' @param max_iter_ica a single numeric that specifies the maximum
iteration of ICA (Default: 1000)
    #' @param type_ica a character that assigns R package between "fastICA"
and "ica" to run independent component analysis (Default: fastICA)
    #' @param lasso_engine a character that assigns R package between
"glmnet" and "lars" to run lasso regression (Default: glmnet)


    max_k = 3,
    max_iter_ica = 1000,
    type_ica = "fastICA",
    lasso_engine = "glmnet",

    causal_order = NULL,
    intercept = NULL,
    adjacency_matrix = NULL,


    initialize = function(
      max_k = 3,
      max_iter_ica = 1000,
      type_ica = "fastICA",
      lasso_engine = "glmnet"){

      self$max_k <- max_k
      self$max_iter_ica <- max_iter_ica
      self$type_ica <- type_ica
      self$lasso_engine <- lasso_engine
    },


    fit = function(X){

      # Specify the number of observed variables
      p <- ncol(X)


      # Impute missing values
      if (length(which(is.na(X))) >= 1){
        X <- missMDA::imputePCA(X, ncp = p) %>%
          .$completeObs %>%
          as.data.frame()
      }
```

```
      history_k <- list()

      while(TRUE){
        for (i_k in c(1:self$max_k)){

          # Estimate unmixing matrix (W_ica) from independent component
analysis

          # Summon fastICA from fastICA package
          # note that W_ica is t(W_ica) of scipy's fastICA
          if(tolower(self$type_ica) == "fastica"){
            ica_out <- X %>% fastICA::fastICA(
              n.comp = p,
              alg.typ = "parallel",
              maxit = self$max_iter_ica,
              fun = "logcosh")
            W_ica <- (ica_out$K %*% ica_out$W)}


          # Summon fastICA from ica package
          if(tolower(self$type_ica) == "icafast"){
            ica_out <- X %>% ica::icafast(
              nc = p,
              alg = "par",
              maxit = self$max_iter_ica,
              fun = "logcosh")
            W_ica <- (t(ica_out$Q) %*% ica_out$R)}


          # Summon information maximum from ica package
          if(tolower(self$type_ica) == "icaimax"){
            ica_out <- X %>% ica::icaimax(
              nc = p,
              maxit = self$max_iter_ica,
              alg = "newton",
              fun = "tanh")
            W_ica <- (t(ica_out$Q) %*% ica_out$R)}


          # Summon JADE from ica package
          if(tolower(self$type_ica) == "icajade"){
            ica_out <- X %>% ica::icajade(
              nc = p,
              maxit = self$max_iter_ica)
            W_ica <- (t(ica_out$Q) %*% ica_out$R)}


          # Permute W_ica
          linassign <- solve_LSAP(1 / abs(t(W_ica)), maximum = FALSE)
          col_index <- unname(linassign)
          PW_ica    <- t(W_ica[,col_index])


          # Scaling vector is the diagonal of permuted W_ica
          D <- diag(PW_ica)


          # Estimate adjacency matrix
          B_ica <- diag(p) - (PW_ica / D)
          colnames(B_ica) <- rownames(B_ica)
```

```
      # Log the estimated causal order
      history_k[[i_k]] <- self$estimate_causal_order(B_ica)
    }

    selected_k <- findListMax(history_k)

    # if no ties happen, endorse the causal order as the finalized
    if(!all(is.na(selected_k))){
      self$causal_order <- selected_k
      break
    }

  }


  # Estimate a matrix of regression coefficients
  reg_out <- X %>% self$estimate_regression_matrix()
  self$intercept <- reg_out$intercept
  self$adjacency_matrix <- reg_out$adjacency_matrix
},


estimate_causal_order = function(B) {
  d <- ncol(B)

  # set m(m + 2) smallest elements (in absolute value) of B to zero
  pos_vec <- order(abs(B))
  initial_zero_num <- d * (d + 1) / 2
  for (i in pos_vec[1:initial_zero_num]) {B[i] <- 0}

  # set to zero until DAG is obtained
  for (i in pos_vec[(initial_zero_num + 1):length(pos_vec)]) {
    B[i] <- 0

    # if B is not DAG, null is returned
    causal_order <- self$search_causal_order(B)
    if (!is.null(causal_order)) {
      break
    }
  }
  return(causal_order)
},


search_causal_order = function(B) {
  causal_order   <- c()
  p              <- nrow(B)
  original_index <- c(1:p)


  while (0 < ncol(B)) {

    # Find a row all of which elements are zero
    row_index_list <- which(rowSums(B) == 0)
    if (length(row_index_list) == 0) break


    # Append ith to the end
    target_index   <- row_index_list[1]
```

```
      causal_order   <- c(causal_order, original_index[target_index])
      original_index <- original_index[-target_index]


      # Remove ith row and ith column
      B <- as.matrix(B[-target_index, -target_index])
    }
    if (!length(causal_order) == p) {
      causal_order <- NULL
    }
    return(causal_order)
  },


  estimate_regression_matrix = function(X) {

    # Intercepts = A
    A <- rep(NA, ncol(X))
    A[self$causal_order[[1]]] <- mean(X[, self$causal_order[[1]]])

    # Regression coefficients matrix = B
    B <- matrix(0, nrow = ncol(X), ncol = ncol(X))


    for (i in 2:length(self$causal_order)) {
      res <- self$predict_adaptive_lasso(X, unlist(self$causal_order[1:(i
- 1)]), unlist(self$causal_order[i]))
      A[unlist(self$causal_order[i])] <- res$intercept
      B[unlist(self$causal_order[i]), unlist(self$causal_order[1:(i -
1)])] <- res$coef
    }

    out <- list()
    out$intercept <- A
    out$adjacency_matrix <- B
    return(out)
  },


  predict_adaptive_lasso = function(X, predictors, target, gamma = 1) {

    # 1st stage (OLS to determine weights)
    fml <- as.formula(paste0(names(X)[target], "~."))
    X_ <- X[names(X)[c(target, predictors)]]
    lr <- lm(fml, data = X_)
    weight <- abs(lr$coefficients[-1])^(gamma)

    # 2nd stage
    x <- as.matrix(X_[, -1, drop = FALSE])
    y <- as.matrix(X_[, 1], drop = FALSE)

    if (tolower(self$lasso_engine) == "lars") {
      # adaptive lasso by lars()
      # use coefs which minimizes bic
      x <- t(t(x) * weight)
      reg <- lars::lars(x = x, y = y, type = "stepwise")
      bic <- log(nrow(x)) * reg$df + nrow(x) * log(reg$RSS/nrow(x))

      # lars does not explicitly return intercept term...
      idx <- which.min(bic)
      coef_ <- matrix(coef(reg), ncol = ncol(x))[idx ,]
```

```
        coef_ <- coef_ * weight
        eval(parse(text = paste0("intercept <- predict(reg, s=", idx, ",
data.frame(", paste0(names(coef_), "=", 0, collapse = ","), "))$fit")))

      } else if (tolower(self$lasso_engine) == "glmnet") {
        # adaptive lasso by glmnet()
        # glmnet() can not handle x with single column

        if (ncol(x) > 1) {
          # specify lambda sequence (default did not search small lambdas)
          lambda_seq <- exp(seq(2, -7, length.out = 80))
          reg <- glmnet::cv.glmnet(x = x, y = y, penalty.factor = 1 /
weight, type.measure = "mse", lambda = lambda_seq, relax = FALSE)

          # use 1se rule
          idx_best  <- which(reg$lambda == reg$lambda.1se)
          coef_     <- reg$glmnet.fit$beta[, idx_best]
          coef_     <- matrix(coef_, ncol = ncol(x))
          intercept <- reg$glmnet.fit$a0[idx_best]
        } else {
          coef_ <- matrix(lr$coefficients[-1], ncol = ncol(x))
          intercept <- lr$coefficients[1]
        }
      }

      # return coefs and intercept
      res <- list()
      res$coef <- coef_
      res$intercept <- intercept
      return(res)
    }
  )
)

#---- DirectLiNGAM ----


DirectLiNGAM <- R6::R6Class(
  "DirectLiNGAM",
  public = list(

    #' @title DirectLiNGAM class
    #' @description To discover the causal relations between observed
variables using DirectLiNGAM algorithm for observed variables
    #' @param lasso_engine a character that specifies the R package to run
lasso regression (Default: glmnet)
    #' @param pk A squared matrix of prior knowledge, with dimension of
factors (q) by factors (q)


    lasso_engine = "glmnet",
    pk = NULL,


    causal_order = NULL,
    partial_orders= NULL,
    intercept = NULL,
    adjacency_matrix = NULL,


    initialize = function(
```

```
      lasso_engine = "glmnet",
      pk = NULL){

      self$lasso_engine <- lasso_engine
      self$pk <- pk

      if (!is.null(self$pk)){self$partial_orders <-
self$extract_partial_orders(self$pk)}
    },


    fit = function(X){

      # Specify the number of observed variables
      p <- ncol(X)

      # Check parameters
      if(is.null(self$pk) == FALSE){
        if(dim(self$pk)[1] != p |
           dim(self$pk)[2] != p){
          sprintf("The shape of prior knowledge must be (%d, %d)", p, p)
        }
      }

      # Causal discovery

      # Index observed variables
      U <- c(1:p)

      # Substanitate an empty list of causal order = K
      K <- c()

      # Define X as X_
      X_ <- X

      for (i_OV in c(1:p)){

        # Determine the first variable = m
        m <- self$search_causal_order(X_, U)

        # Remove the effect of m from the rest of U
        for (i in U){
          if (i != m){
            X_[,i] = self$residual(X_[,i], X_[,m])
          }
        }

        # Variable m enters the end of K
        K <- c(K, m)

        # Drop m from U
        U <- U[U != m]

        # Update partial orders
        # if (!is.null(self$pk)){
        #   self$partial_orders <-
self$partial_orders[which(self$partial_orders[,1]!=m),]
        # }
      }

      self$causal_order <- K
```

```
      # Estimate the regression coefficient matrix
      reg_out <- X %>% self$estimate_regression_matrix()
      self$intercept <- reg_out$intercept
      self$adjacency_matrix <- reg_out$adjacency_matrix
    },

    extract_partial_orders = function(pk){

      # Extract partial orders from prior knowledge.
      pos_path_pair <- which(pk == 1)
      pos_nopath_pair <- which(pk == 0)

      # Check for inconsistencies in pairs with path
      check_path_pair <- c()
      for (i in c(1:length(pos_path_pair))){
        tIndex <- pos_path_pair[i]
        if (tIndex %% ncol(pk) == 0){
          tRow <- ncol(pk)
          tCol <- tIndex %/% ncol(pk)
        } else {
          tRow <- tIndex %% ncol(pk)
          tCol <- tIndex %/% ncol(pk) + 1
        }
        check_path_pair <- rbind(check_path_pair,c(tCol, tRow))
        check_path_pair <- rbind(check_path_pair,c(tRow, tCol))
      }

      if(sum(duplicated(check_path_pair) >= 1)){
        cat("The prior knowledge contains inconsistencies (Col, Row):\n")
        message(check_path_pair[duplicated(check_path_pair),])
      }

      path_pair <- check_path_pair[seq(1,nrow(check_path_pair),2),]

      # Check for inconsistencies in pairs without path
      # If there are duplicate pairs without path, they cancel out and are
not ordered.
      check_nopath_pair <- list()
      for (i in c(1:length(pos_nopath_pair))){
        tIndex <- pos_nopath_pair[i]
        if (tIndex %% ncol(pk) == 0){
          tRow <- ncol(pk)
          tCol <- tIndex %/% ncol(pk)
        } else {
          tRow <- tIndex %% ncol(pk)
          tCol <- tIndex %/% ncol(pk) + 1
        }
        check_nopath_pair <- rbind(check_path_pair,c(tCol, tRow))
        check_nopath_pair <- rbind(check_path_pair,c(tRow, tCol))
      }

      nopath_pair <- unique(check_nopath_pair)

      all_pairs <- rbind(path_pair, nopath_pair)

      if (nrow(all_pairs) == 0) {
        # if no pairs are extracted from the specified prior knowledge
        # discard the prior knowledge.
        self$pk = NULL
        return(NULL)
```

```
  }

  all_pairs <- unique(all_pairs)
  return(all_pairs)
},

residual = function(xi, xj){
  # The residual when xi is regressed on xj.
  xi - (cov(xi, xj) / var(xj)) * xj
},

entropy = function(u){
  # Calculate entropy using the maximum entropy approximation
  k1 <- 79.047
  k2 <- 7.4129
  gamma <- 0.37457
  return(
    (1 + log(2 * pi)) / 2 -
      k1 * (mean(log(cosh(u))) - gamma)^2 -
      k2 * (mean(u * exp((-u^2) / 2)))^2)
},

diff_mutual_info = function(xi_std, xj_std, ri_j, rj_i){
  # Calculate the difference of the mutual informations.
  term1 <- self$entropy(xj_std) + self$entropy(ri_j / sd(ri_j))
  term2 <- self$entropy(xi_std) + self$entropy(rj_i / sd(rj_i))
  return(term1 - term2)
},

search_candidate = function(U){
  # Search for candidate features.

  # If no prior knowledge is specified, nothing to do.
  if (is.null(self$pk)){
    out <- c()
    out$Uc <- U
    out$Vj <- list()
    return(out)
  }

  # Find exogenous features
  Uc = c()
  for (j in U){
    index = U[U != j]
    if(sum(self$pk[j,index]) == 0){Uc <- c(Uc, j)}
  }

  # Find endogenous features, and then find candidate features
  if (length(Uc) == 0){
    U_end <- c()
    for (j in U){
      index <- U[U != j]
      if(sum(self$pk[j,index],na.rm=T) > 0){U_end <- c(U_end, j)}
    }

    # Find sink features (original)
    for (i in U){
      index <- U[U != i]
      if(sum(self$pk[index, i]) == 0){U_end <- c(U_end, i)}
    }
    Uc <- setdiff(U, U_end)
```

```
    }

    # make V^(j)
    Vj <- c()
    for (i in U){
      if (i %in% Uc){
        if (sum(self$pk[i,Uc]==0)){Vj <- c(Vj, i)}
      }
    }
    out <- c()
    if (is.null(Uc)){out$Uc <- U}else{out$Uc <- c(Uc)}
    out$Vj <- c(Vj)
    return(out)
  },

  search_causal_order = function(X, U){
    # Search the causal ordering.
    sc_out <- self$search_candidate(U)
    Uc <- sc_out$Uc
    Vj <- sc_out$Vj

    if (length(Uc) == 1){return(Uc[1])}

    M_list <- c()
    for (i in Uc){
      M = 0
      for (j in U){
        if (i != j){
          xi_std = (X[, i] - mean(X[, i])) / sd(X[, i])
          xj_std = (X[, j] - mean(X[, j])) / sd(X[, j])
          if (i %in% Vj & j %in% Uc){
            ri_j <- xi_std
          } else {ri_j <- self$residual(xi_std, xj_std)}

          if (j %in% Vj & i %in% Uc){
            rj_i <- xj_std
          } else {rj_i <- self$residual(xj_std, xi_std)}

          M = M + min(0,self$diff_mutual_info(xi_std, xj_std, ri_j,
rj_i))^2
        }
      }
      M_list <- c(M_list, -M)
    }
    return(Uc[which.max(M_list)])
  },

  mutual_information = function(x1, x2, param){

    # Calculate the mutual informations.
    kappa <- param[1]; sigma <- param[2]
    n <- length(x1)
    X1 <- matrix(c(rep(x1,n)),nrow = n, ncol = n, byrow = T)
    K1 <- exp(-1 / (2*sigma^2) * (X1^2 + t(X1)^2 - 2 * X1 * t(X1)))
    X2 <- matrix(c(rep(x2,n)),nrow = n, ncol = n, byrow = T)
    K2 <- exp(-1 / (2*sigma^2) * (X2^2 + t(X2)^2 - 2 * X2 * t(X2)))

    tmp1 <- K1 + (n * kappa * diag(n) / 2)
    tmp2 <- K2 + (n * kappa * diag(n) / 2)

    m1 <- (tmp1 %*% tmp1)
```

```
      m2 <- (K1 %*% K2)
      m3 <- (K2 %*% K1)
      m4 <- (tmp2 %*% tmp2)
      m_zero <- matrix(0,nrow=n,ncol=n)

      K_kappa <- rbind(
        cbind(m1, m2),
        cbind(m3, m4))
      D_kappa <- rbind(
        cbind(m1, m_zero),
        cbind(m_zero, m4))

      sigma_K <- svd(K_kappa)$d
      sigma_D <- svd(D_kappa)$d

      return((-1/2)*(sum(log(sigma_K), na.rm=T) - sum(log(sigma_D),
na.rm=T)))
    },

    search_causal_order_kernel = function(X, U){
      # Search the causal ordering by kernel method
      sc_out <- self$search_candidate(U = U)
      Uc <- sc_out$Uc;    Vj <- sc_out$Vj
      if (length(Uc)==1){
        return(Uc[1])}

      if (nrow(X) > 1000){
        param = c(2e-3, 0.5)
      }else{
        param = c(2e-2, 1.0)}

      Tkernels <- c()
      for (j in Uc){
        Tkernel <- 0
        for (i in U){
          if (i != j){
            if (j %in% Vj & i %in% Uc){
              ri_j <- X[,i]
            } else {
              ri_j <- residual(X[,i], X[,j])
            }
            Tkernel = Tkernel + self$mutual_information(X[,j], ri_j, param)
          }
        }
        Tkernels <- c(Tkernels, Tkernel)
      }

      return(Uc[which.min(Tkernels)])
    },

    estimate_regression_matrix = function(X) {

      # Intercept = A
      A <- rep(NA, ncol(X))
      A[self$causal_order[[1]]] <- mean(X[, self$causal_order[[1]]])

      # Regression coefficient matrix = B
      B <- matrix(0, nrow = ncol(X), ncol = ncol(X))

      for (i in 2:length(self$causal_order)) {
```

```
        res <- self$predict_adaptive_lasso(X, unlist(self$causal_order[1:(i
- 1)]), unlist(self$causal_order[i]))
        A[unlist(self$causal_order[i])] <- res$intercept
        B[unlist(self$causal_order[i]), unlist(self$causal_order[1:(i -
1)])] <- res$coef
      }

      out <- list()
      out$intercept <- A
      out$adjacency_matrix <- B
      return(out)
    },

    predict_adaptive_lasso = function(X, predictors, target, gamma = 1) {

      # 1st stage (OLS to determine weights)
      X <- as.data.frame(X)
      fml <- as.formula(paste0(names(X)[target], "~."))
      X_ <- X[names(X)[c(target, predictors)]]
      lr <- lm(fml, data = X_)
      weight <- abs(lr$coefficients[-1])^(gamma)

      # 2nd stage
      x <- as.matrix(X_[, -1, drop = FALSE])
      y <- as.matrix(X_[, 1], drop = FALSE)

      if (self$lasso_engine == "lars") {
        # adaptive lasso by lars()
        # use coefs which minimizes bic
        x <- t(t(x) * weight)
        reg <- lars::lars(x = x, y = y, type = "stepwise")
        bic <- log(nrow(x)) * reg$df + nrow(x) * log(reg$RSS/nrow(x))

        # lars does not explicitly return intercept term...
        idx <- which.min(bic)
        coef_ <- matrix(coef(reg), ncol = ncol(x))[idx ,]
        coef_ <- coef_ * weight
        eval(parse(text = paste0("intercept <- predict(reg, s=", idx, ",
data.frame(", paste0(names(coef_), "=", 0, collapse = ","), "))$fit")))

      } else if (self$lasso_engine == "glmnet") {
        # adaptive lasso by glmnet()
        # glmnet() can not handle x with single column

        if (ncol(x) > 1) {
          # specify lambda sequence (default did not search small lambdas)
          lambda_seq <- exp(seq(2, -7, length.out = 80))
          reg <- glmnet::cv.glmnet(x = x, y = y, penalty.factor = 1 /
weight, type.measure = "mse", lambda = lambda_seq, relax = FALSE)

          # use 1se rule
          idx_best  <- which(reg$lambda == reg$lambda.1se)
          coef_     <- reg$glmnet.fit$beta[, idx_best]
          coef_     <- matrix(coef_, ncol = ncol(x))
          intercept <- reg$glmnet.fit$a0[idx_best]
        } else {
          coef_ <- matrix(lr$coefficients[-1], ncol = ncol(x))
          intercept <- lr$coefficients[1]
        }
      }
```

```
      # return coefs and intercept
      res <- list()
      res$coef <- coef_
      res$intercept <- intercept
      return(res)
    }

  )
)

#---- ParceLiNGAM ----


ParceLiNGAM <- R6::R6Class(
  "ParceLiNGAM",
  public = list(

    #' @title DirectLiNGAM class
    #' @description To discover the causal relations between observed
variables using ParceLiNGAM algorithm for observed variables
    #' @param lasso_engine a character that specifies the R package to run
lasso regression (Default: glmnet)
    #' @param alpha a numeric that specifies a uncorrected threshold value
for Fisher's independence test (Default: .0011)
    #' @param pk A squared matrix of prior knowledge, with dimension of
factors (q) by factors (q)


    lasso_engine = "glmnet",
    alpha = 0.001,
    pk = NULL,

    causal_order = NULL,
    partial_orders = NULL,
    list_pHSIC = NULL,
    intercept = NULL,
    adjacency_matrix = NULL,


    initialize = function(
      lasso_engine = "glmnet",
      alpha = 0.001,
      pk = NULL){

      self$pk <- pk
      self$lasso_engine <- lasso_engine
      self$alpha <- alpha

      if (!is.null(self$pk)){self$partial_orders <-
self$extract_partial_orders(self$pk)}
    },


    fit = function(X){

      # Specify the number of observed variables = p
      p <- ncol(X)

      # Check prior knowledge
      if(!is.null(self$pk)){
        if(dim(self$pk)[1] != p |
```

```
          dim(self$pk)[2] != p){
        sprintf("The shape of prior knowledge must be (%d, %d)", p, p)
      }
    }

    # Centering
    X <- scale(X)


    # Apply bonferroni correction
    thresh_p <- self$alpha / (p-1)


    # Search causal orders one by one from the bottom upward
    sc_out <- self$search_causal_order(X, thresh_p)
    self$causal_order <- sc_out$K_bttm
    self$list_pHSIC <- sc_out$p_bttm

    # Estimate the regression coefficient matrix
    reg_out <- X %>% self$estimate_regression_matrix()
    self$intercept <- reg_out$intercept
    self$adjacency_matrix <- reg_out$adjacency_matrix

  },


  extract_partial_orders = function(pk){

    # Extract partial orders from prior knowledge.
    pos_path_pair <- which(pk == 1)
    pos_nopath_pair <- which(pk == 0)

    # Check for inconsistencies in pairs with path
    check_path_pair <- c()
    for (i in c(1:length(pos_path_pair))){
      tIndex <- pos_path_pair[i]
      if (tIndex %% ncol(pk) == 0){
        tRow <- ncol(pk)
        tCol <- tIndex %/% ncol(pk)
      } else {
        tRow <- tIndex %% ncol(pk)
        tCol <- tIndex %/% ncol(pk) + 1
      }
      check_path_pair <- rbind(check_path_pair,c(tCol, tRow))
      check_path_pair <- rbind(check_path_pair,c(tRow, tCol))
    }

    if(sum(duplicated(check_path_pair) >= 1)){
      cat("The prior knowledge contains inconsistencies (Col, Row):\n")
      message(check_path_pair[duplicated(check_path_pair),])
    }

    path_pair <- check_path_pair[seq(1,nrow(check_path_pair),2),]

    # Check for inconsistencies in pairs without path
    # If there are duplicate pairs without path, they cancel out and are
not ordered.
    check_nopath_pair <- list()
    for (i in c(1:length(pos_nopath_pair))){
      tIndex <- pos_nopath_pair[i]
      if (tIndex %% ncol(pk) == 0){
```

```
      tRow <- ncol(pk)
      tCol <- tIndex %/% ncol(pk)
    } else {
      tRow <- tIndex %% ncol(pk)
      tCol <- tIndex %/% ncol(pk) + 1
    }
    check_nopath_pair <- rbind(check_path_pair,c(tCol, tRow))
    check_nopath_pair <- rbind(check_path_pair,c(tRow, tCol))
  }

  nopath_pair <- unique(check_nopath_pair)

  all_pairs <- rbind(path_pair, nopath_pair)

  if (nrow(all_pairs) == 0) {
    # if no pairs are extracted from the specified prior knowledge
    # discard the prior knowledge.
    self$pk = NULL
    return(NULL)
  }

  all_pairs <- unique(all_pairs)
  return(all_pairs)
},


residual = function(xi, xj){
  # The residual when xi is regressed on xj.
  xi - (cov(xi, xj) / var(xj)) * xj
},


search_candidate = function(U){
  # Search for candidate features
  # If no prior knowledge is specified, nothing to do.
  if (is.null(self$pk)){
    return(U)}

  # Candidate features that are not to the left of the partial order
  diff_U <- unlist(self$partial_orders[,1])
  Uc <- setdiff(U, diff_U)
  return(Uc)
},


search_causal_order = function(X, thresh_p){

  # Search causal orders one by one from the bottom upward.

  # Index observed variables
  U <- c(1:ncol(X))

  # Substantiate empty lists of K_bttm and p_bttm
  K_bttm <- c()
  p_bttm <- c()


  is_search_causal_order <- TRUE

  # If only one variable is left in list U, enter the top of K_bttm
  if(length(U) <= 1){
```

```
        K_bttm <- c(U, K_bttm)
        p_bttm <- c(0, p_bttm)
        is_search_causal_order <- FALSE}


      while (is_search_causal_order) {

        # Search for candidate features
        Uc <- self$search_candidate(U)


        if (length(Uc) == 1){

          # If there is only one variable in Uc,
          # calculate HSIC with the rest of the variables
          m <- c(Uc[1])
          predictors <- setdiff(U, Uc[1])
          res <- self$compute_residuals(X, predictors, m)
          fisher_p <- self$fisher_hsic_test(X[,predictors],res,Inf)[1]

        } else {

          # Find the most sink variable
          exo_out <- self$find_exo_vec(X, Uc)

          # Index of the sink variable = m
          m <- as.numeric(exo_out["m"])

          # p-value of HSIC with Fisher's method = fisher_p
          fisher_p <- as.numeric(exo_out["max_p"])
        }

        # Conduct statistical test by the p-value or the statistic
        # If statistical test is not rejected
        if (fisher_p >= thresh_p) {


          # Add index of the exogenous variable to the top of K_bttm
          K_bttm <- c(m, K_bttm)

          # Add fisher's p-value of the sink variable to the top of p_bttm
          p_bttm <- c(fisher_p, p_bttm)


          # Update U by dropping the index of the sink variable (m)
          U <- U[U != m]


          # Update the partial order
          if (!is.null(self$pk)){
            self$partial_orders <-
self$partial_orders[which(self$partial_orders[,1]!=m),]
          }


          # If there is only one candidate for sink variable, end the
search
          if (length(U) <= 1){
            K_bttm <- c(U, K_bttm)
            p_bttm <- c(0, p_bttm)
            is_search_causal_order <- FALSE}
```

```
      } else {

        # If statistical test is rejected
        is_search_causal_order <- FALSE
      }
    }

    out <- c()
    out$K_bttm <- K_bttm
    out$p_bttm <- p_bttm
    return(out)
  },


  find_exo_vec = function(X, U){
    # Find the most exogenous vector.
    max_p <- -Inf
    max_p_stat <- Inf

    exo_vec <- c()
    for (j in (1:length(U))){
      xi_index <- setdiff(U,U[j])
      xj_index <- c(U[j])

      # Compute residuals
      res <- self$compute_residuals(X, xi_index, xj_index)

      # HSIC test with Fisher's method to test the independence between
      # a set of predictors (xi_index) and error term (res)
      fisher_hsic_out <- self$fisher_hsic_test(
        X[,xi_index],res,max_p_stat)

      # Retrieve the p-value of HSIC with Fisher's method = fisher_p
      fisher_p <- fisher_hsic_out[1]

      # Retrieve the statistic of HSIC with Fisher's method = fisher_stat
      fisher_stat <- fisher_hsic_out[2]

      # If the fisher_p of j is the greatest so far, update
      # the index of explanatory variables (exo_vec),
      # the maximum of fisher_p (max_p), and;
      # the maximum of fisher_stat (max_p_stat)
      if (fisher_stat < max_p_stat | fisher_p > max_p){
        exo_vec <- xi_index
        max_p <- fisher_p
        max_p_stat <- fisher_stat
      }
    }

    # Obtain the index of sink variable that is not included in exo_vec =
m
    m <- setdiff(U, exo_vec)
    out <- c()
    out <- c(m, exo_vec, max_p) %>%
      'names<-'(c("m","exo_vec","max_p"))
    return(out)
  },


  fisher_hsic_test = function(X, res, max_p_stat){
```

```
      # Conduct statistical test by HSIC with Fisher's methods.

      # Substantiate the statistic of HSIC with Fisher's method =
fisher_stat
      fisher_stat <- 0


      if (is.null(ncol(X))){
        # If X is a vector, set q = 1
        q <- 1
      } else {
        # If X is a matrix, set q = its columm length
        q <- ncol(X)}

      if (q == 1){
        # If there is only one predictor, simply HSIC will do
        hsic_out <- dHSIC::dhsic.test(
          X = X,
          Y = res,
          kernel = "gaussian",
          method = "gamma")

        # Retrieve p-value of HSIC (without Fisher's method) = fisher_p
        fisher_p <- hsic_out$p.value

        # Retrieve the statistic of HSIC (without Fisher's method) =
fisher_stat
        fisher_stat <- hsic_out$statistic

      } else {

        # If there is more than one predictor, turn HSIC with Fisher's
method
        for (i in c(1:q)){

          # Conduct pairwise HSIC and retrieve the p-value of HSIC
          hsic_p <- dHSIC::dhsic.test(
            X = as.data.frame(X[,i]),
            Y = res,
            kernel = "gaussian",
            method = "gamma")$p.value


          if (hsic_p == 0){
            # If p-value of HSIC is zero, set fisher_stat as infinite
            fisher_stat <- Inf
          } else {

            # If p-value of HSIC is non-zero, add to fisher_stat
            fisher_stat <- fisher_stat + (-2 * log(hsic_p))
          }

          if (fisher_stat > max_p_stat){break}
        }

        # Compute the p-value of HSIC with Fisher's method by doing chi-
square test
        fisher_p <- pchisq(fisher_stat, df = (2 * q), lower.tail=FALSE)
      }
      return(c(fisher_p, fisher_stat) %>% 'names<-
'(c("fisher_p","fisher_stat")))
```

```
    },


    compute_residuals = function(X, predictors, target){
      # Compute residuals
      if (is.null(self$reg)){
        # Compute residuals of least square regressions
        cov <- cov((X))
        term1 <- MASS::ginv(cov[predictors,predictors])
        term2 <- matrix(
          cov[target, predictors],
          nrow = length(predictors),
          ncol = 1)
        coef <- term1 %*% term2
        res <- as.matrix(X[,target]) - as.matrix(X[,predictors]) %*% coef
      } else {
        lm_fit <- lm(X[,target] ~., data = X[,predictors])
        res <- lm_fit$residuals
      }

      return(res)
    },


    estimate_regression_matrix = function(X) {

      # Intercept = A
      A <- rep(NA, ncol(X))
      A[self$causal_order[[1]]] <- mean(X[, self$causal_order[[1]]])

      # Regression coefficient matrix = B
      B <- matrix(0, nrow = ncol(X), ncol = ncol(X))

      for (i in 2:length(self$causal_order)) {
        res <- self$predict_adaptive_lasso(X, unlist(self$causal_order[1:(i
- 1)]), unlist(self$causal_order[i]))
        A[unlist(self$causal_order[i])] <- res$intercept
        B[unlist(self$causal_order[i]), unlist(self$causal_order[1:(i -
1)])] <- res$coef
      }

      out <- list()
      out$intercept <- A
      out$adjacency_matrix <- B
      return(out)
    },


    predict_adaptive_lasso = function(X, predictors, target, gamma = 1) {

      # 1st stage (OLS to determine weights)
      X <- as.data.frame(X)
      fml <- as.formula(paste0(names(X)[target], "~."))
      X_ <- X[names(X)[c(target, predictors)]]
      lr <- lm(fml, data = X_)
      weight <- abs(lr$coefficients[-1])^(gamma)

      # 2nd stage
      x <- as.matrix(X_[, -1, drop = FALSE])
      y <- as.matrix(X_[, 1], drop = FALSE)
```

```
      if (self$lasso_engine == "lars") {
        # adaptive lasso by lars()
        # use coefs which minimizes bic
        x <- t(t(x) * weight)
        reg <- lars::lars(x = x, y = y, type = "stepwise")
        bic <- log(nrow(x)) * reg$df + nrow(x) * log(reg$RSS/nrow(x))

        # lars does not explicitly return intercept term...
        idx <- which.min(bic)
        coef_ <- matrix(coef(reg), ncol = ncol(x))[idx ,]
        coef_ <- coef_ * weight
        eval(parse(text = paste0("intercept <- predict(reg, s=", idx, ",
data.frame(", paste0(names(coef_), "=", 0, collapse = ","), "))$fit")))

      } else if (self$lasso_engine == "glmnet") {
        # adaptive lasso by glmnet()
        # glmnet() can not handle x with single column

        if (ncol(x) > 1) {
          # specify lambda sequence (default did not search small lambdas)
          lambda_seq <- exp(seq(2, -7, length.out = 80))
          reg <- glmnet::cv.glmnet(x = x, y = y, penalty.factor = 1 /
weight, type.measure = "mse", lambda = lambda_seq, relax = FALSE)

          # use 1se rule
          idx_best  <- which(reg$lambda == reg$lambda.1se)
          coef_     <- reg$glmnet.fit$beta[, idx_best]
          coef_     <- matrix(coef_, ncol = ncol(x))
          intercept <- reg$glmnet.fit$a0[idx_best]
        } else {
          coef_ <- matrix(lr$coefficients[-1], ncol = ncol(x))
          intercept <- lr$coefficients[1]
        }
      }

      # return coefs and intercept
      res <- list()
      res$coef <- coef_
      res$intercept <- intercept
      return(res)
    }

  )
)

#----Observed-Variable RCD ----

#' @title RCD_OV class
#' @description R implementation of repetitive causal discovery for latent
variables
#' @references T.N.Maeda and S.Shimizu. RCD: Repetitive causal discovery of
linear non-Gaussian acyclic models with latent confounders. In Proc. 23rd
International Conference on Artificial Intelligence and Statistics
(AISTATS2020), Palermo, Sicily, Italy. PMLR  108:735-745, 2020.
#' @export

RCD_OV <- R6::R6Class(
  "RCD_OV",
  public = list(

    random_state = NULL,
```

```
max_explanatory_num = 2,
cor_alpha = 0.01,
ind_alpha = 0.01,
shapiro_alpha = 0.01,
lasso_engine = "glmnet",

adjacency_matrix = NULL,
ancestors_list = NULL,
parents_list = NULL,

initialize = function(
  random_state = NULL,
  max_explanatory_num = 2,
  cor_alpha = 0.01,
  ind_alpha = 0.01,
  shapiro_alpha = 0.01,
  lasso_engine = "glmnet"){

  self$random_state <- random_state
  self$max_explanatory_num <- max_explanatory_num
  self$cor_alpha = cor_alpha
  self$ind_alpha = ind_alpha
  self$shapiro_alpha = shapiro_alpha
  self$lasso_engine = lasso_engine

  if(max_explanatory_num <= 0){
    warning("Maximum number of explanatory variable must be >0.")}
  if(cor_alpha < 0){
    warning("cor_alpha must be >=0.")
  }
  if(ind_alpha <0){
    warning("ind_alpha must be >=0.")
  }
  if(shapiro_alpha <0){
    warning("shapiro_alpha must be >=0.")
  }
},

fit = function(X){

  # Check parameters
  n_features <- ncol(X)

  # Extract a set of ancestors of each variable
  M <- self$extract_ancestors(X)

  # Extract parents (direct causes) from the set of ancestors.
  P <- self$extract_parents(X, M)

  # Find the pairs of variables affected by the same latent confounders
  C <- self$extract_vars_sharing_confounders(X, P)

  self$parents_list <- P
  self$ancestors_list <- M
  self$adjacency_matrix <- self$estimate_adjacency_matrix(X, P, C)
},

get_common_ancestors = function(M, U){
  # Get the set of common ancestors of U
  Mj_list <- list(); c = 0
  for (xj in U){
```

```
        c = c + 1; Mj_list[[c]] <- M[[xj]]
      }

      out <- Reduce(intersect, Mj_list)
      if(length(out) == 0){return(NA)} else {
        return(Reduce(intersect, Mj_list))
      }
    },

    get_resid_and_coef = function(X, endog_idx, exog_idcs){
      # Get the residuals and coefficients of the ordinary least square
method
      X <- as.data.frame(X)
      fml <- as.formula(paste0(names(X)[endog_idx], "~."))
      X_ <- X[names(X)[c(endog_idx, exog_idcs)]]
      lr <- lm(fml, data = X_)
      out <- list()
      out$resid <- lr$residuals
      out$coef <- lr$coefficients
      return(out)
    },

    get_residual_matrix = function(X, U, H_U){
      if (length(H_U) == 1 & all(is.na(H_U)) == TRUE){return(X)}
      if (all(is.na(H_U))){return(X)}

      Y <- X
      Y[,] <- 0
      for (xj in U){
        Y[,xj] <- self$get_resid_and_coef(X,xj,H_U)$resid
      }
      return(Y)
    },

    is_non_gaussianity = function(Y, U){
      # Test whether a variable is generated from a non-Gaussian process
using the Shapiro-Wilk test
      for (xj in U){
        if (shapiro.test(Y[,xj])$p.value >
self$shapiro_alpha){return(FALSE)}
      }
      return(TRUE)
    },

    is_correlated = function(a, b){
      # Estimate that the two variables are linearly correlated
      return(Hmisc::rcorr(a,b,type = c("pearson"))$P[1,2] < self$cor_alpha)
    },

    exists_ancestor_in_U = function(M,U,xi,xj_list){
      # Check if xi is not in Mj, the ancestor of xj.
      for (xj in xj_list){
        if (xi %in% M[[xj]]){return(TRUE)}
      }

      # Check if xj_list is a subset of Mi, the ancestor of xi.
      if (all(is.na(M[[xi]])) == FALSE){
        if (all(xj_list == intersect(xj_list, M[[xi]]))){return(TRUE)}
      }
      return(FALSE)
    },
```

```
    is_independent = function(X,Y){
      fisher_p <- dHSIC::dhsic.test(
        X, Y, method = "gamma", kernel = "gaussian")$p.value
      return(fisher_p > self$ind_alpha)
    },

    is_independent_of_resid = function(Y, xi, xj_list){
      # Check whether the residuals obtained from multiple regression
      n_samples <- nrow(Y)

      # Multiple Regression with OLS.
      is_all_independent <- TRUE
      resid <- self$get_resid_and_coef(Y,xi,xj_list)$resid
      for (xj in xj_list){
        if (self$is_independent(resid, Y[,xj]) == FALSE){
          is_all_independent <- FALSE
          break
        }
      }

      return(is_all_independent)
    },

    extract_ancestors = function(X){
      # Extract a set of ancestors of each variable
      n_features <- ncol(X)
      M <- list()
      for (i in c(1:n_features)){M[[i]] <- c(NA)}
      l <- 1
      hu_history <- list()

      while(TRUE){
        changed <- FALSE
        U_list <- combn(c(1:n_features),l+1,simplify = FALSE)
        for (U in U_list){
          U <- sort(U)

          # Get the set of common ancestors of U
          H_U <- self$get_common_ancestors(M, U)

          str_U <- as.character(paste(U,collapse = " "))

          if (is.null(hu_history[[str_U]]) == FALSE){
            if (all(is.na(H_U))==FALSE &
all(is.na(hu_history[[str_U]]))==FALSE){
              if (identical(H_U, hu_history[[str_U]])) {next}
            }
          }

          Y <- as.data.frame(self$get_residual_matrix(X, U, H_U))

          # Test whether a variable is generated from a non-Gaussian
process using the Shapiro-Wilk test
          if (self$is_non_gaussianity(Y,U) == FALSE){next}

          # Estimate that the two variables are linearly correlated using
the Pearson's Correlation
          is_cor <- TRUE
          for (i in combn(U,2, simplify = FALSE)){
            xi <- i[1]; xj <- i[2]
```

```
            if (self$is_correlated(Y[,xi], Y[,xj]) == FALSE){
              is_cor <- FALSE
              break
            }
          }
          if (is_cor == FALSE){next}

          sink_set <- c()
          for (xi in U){
            xj_list <- setdiff(U,xi)
            if(self$exists_ancestor_in_U(M,U,xi,xj_list)){next}

            # Check whether the residuals obtained from multiple
regressions are independent
            if(self$is_independent_of_resid(Y,xi,xj_list)){
              sink_set <- c(sink_set, xi)
            }
          }

          if (length(sink_set)==1){
            xi <- sink_set[1]
            xj_list <- setdiff(U,xi)

            if (all(is.na(M[[xi]]))){
              M[[xi]] <- xj_list
              changed <- TRUE
            }

            if (all(is.na(M[[xi]])) == FALSE){
              if(identical(M[[xi]], union(M[[xi]], xj_list)) == FALSE){
                M[[xi]] <- union(M[[xi]], xj_list)
                changed <- TRUE
              }
            }
          }

          hu_history[[str_U]] <- H_U
        }

        if (changed){
          l <- 1
        } else if (l < self$max_explanatory_num){
          l  <- l + 1
        } else {
          break
        }
      }
      return(M)
    },

    is_parent = function(X, M, xj, xi){
      if (length(setdiff(M[[xi]], xj)) > 0){
        zi <- self$get_resid_and_coef(X,xi,setdiff(M[[xi]],xj))$resid
      } else {
        zi <- X[,xi]
      }

      if (length(intersect(M[[xi]], M[[xj]])) > 0){
        wj <-
self$get_resid_and_coef(X,xj,intersect(M[[xi]],M[[xj]]))$resid
      } else {
```

```
    wj <- X[,xj]
  }

  # Check if zi and wj are correlated
  return(self$is_correlated(wj, zi))
},

extract_parents = function(X, M){
  # Extract parents (direct causes) from a set of ancestors.
  n_features <- ncol(X)
  P <- list()
  for (i in c(1:n_features)){P[[i]] <- c(NA)}

  for (xi in c(1:n_features)){
    if (all(is.na(M[[xi]]))==FALSE){
      for (xj in M[[xi]]){
        # Check if xj is the parent of xi.
        if (self$is_parent(X, M, xj, xi) == TRUE){
          P[[xi]] <- unique(c(P[[xi]], xj))
          P[[xi]] <- P[[xi]][!is.na(P[[xi]])]
        }
      }
    }
  }
  return(P)
},

get_resid_to_parent = function(X,idx,P){
  if (all(is.na(P[[idx]]))){return(X[,idx])}
  return(self$get_resid_and_coef(X,idx,c(P[[idx]]))$resid)
},

extract_vars_sharing_confounders = function(X,P){
  # Find the pairs of variables affect by the same latent confounders.
  n_features <- ncol(X)
  C <- list()
  for (i in c(1:n_features)){C[[i]] <- c(NA)}

  for (k in combn(c(1:n_features),2, simplify = FALSE)){
    i <- k[1]; j <- k[2]
    if (i %in% P[[j]] | j %in% P[[i]]){
      next
    }
    resid_xi <- self$get_resid_to_parent(X, i, P)
    resid_xj <- self$get_resid_to_parent(X, j, P)
    if (self$is_correlated(resid_xi, resid_xj)){
      C[[i]] <- unique(c(C[[i]],j))
      C[[j]] <- unique(c(C[[j]],i))
      C[[i]] <- C[[i]][!is.na(C[[i]])]
      C[[j]] <- C[[j]][!is.na(C[[j]])]

    }
  }
  return(C)
},

estimate_adjacency_matrix = function(X, P, C) {
  # Estimate adjacency matrix by causal parents and confounders

  # Check parents
  n_features <- ncol(X)
```

```
      B <- matrix(0, nrow = n_features, ncol = n_features)
      for (xi in c(1:n_features)){
        xj_list <- P[[xi]]
        if (all(is.na(xj_list))){next}
        xj_list <- sort(xj_list)
        B[xi,c(xj_list)] <- self$predict_adaptive_lasso(X,xj_list,xi)$coef
      }

      # Check confounders
      for (xi in c(1:n_features)){
        xj_list <- C[[xi]]
        if (all(is.na(xj_list))){next}
        xj_list <- sort(xj_list)

        for (xj in xj_list){
          B[xi,xj] <- NA
        }
      }
      return(B)
    },

    predict_adaptive_lasso = function(X, predictors, target, gamma = 1) {

      # 1st stage (OLS to determine weights)
      fml <- as.formula(paste0(names(X)[target], "~."))
      X_ <- X[names(X)[c(target, predictors)]]
      lr <- lm(fml, data = X_)
      weight <- abs(lr$coefficients[-1])^(gamma)

      # 2nd stage
      x <- as.matrix(X_[, -1, drop = FALSE])
      y <- as.matrix(X_[, 1], drop = FALSE)

      if (self$lasso_engine == "lars") {
        # adaptive lasso by lars()
        # use coefs which minimizes bic
        x <- t(t(x) * weight)
        reg <- lars::lars(x = x, y = y, type = "stepwise")
        bic <- log(nrow(x)) * reg$df + nrow(x) * log(reg$RSS/nrow(x))

        # lars does not explicitly return intercept term...
        idx <- which.min(bic)
        coef_ <- matrix(coef(reg), ncol = ncol(x))[idx ,]
        coef_ <- coef_ * weight
        eval(parse(text = paste0("intercept <- predict(reg, s=", idx, ",
data.frame(", paste0(names(coef_), "=", 0, collapse = ","), "))$fit")))

      } else if (self$lasso_engine == "glmnet") {
        # adaptive lasso by glmnet()
        # glmnet() can not handle x with single column

        if (ncol(x) > 1) {
          # specify lambda sequence (default did not search small lambdas)
          lambda_seq <- exp(seq(2, -7, length.out = 80))
          reg <- glmnet::cv.glmnet(x = x, y = y, penalty.factor = 1 /
weight, type.measure = "mse", lambda = lambda_seq, relax = FALSE)

          # use 1se rule
          idx_best  <- which(reg$lambda == reg$lambda.1se)
          coef_     <- reg$glmnet.fit$beta[, idx_best]
          coef_     <- matrix(coef_, ncol = ncol(x))
```

```
          intercept <- reg$glmnet.fit$a0[idx_best]
        } else {
          coef_ <- matrix(lr$coefficients[-1], ncol = ncol(x))
          intercept <- lr$coefficients[1]
        }
      }

      # return coefs and intercept
      res <- list()
      res$coef <- coef_
      res$intercept <- intercept
      return(res)
    }


  )
)



#---- Plot ----

#' Visualize graph based on adjacency matrix
#' @description calls DiagrammeR
#' @param adj_mat (numerical matrix) from: col -> to: row
#' @param node_labels (numeric or character vector) node labels
#' @param round_digit (integer) maximum digits
#' @import DiagrammeR
#' @export

plot_adjacency_mat <- function(adj_mat, node_labels = NULL, round_digit =
2) {
  nodes <- DiagrammeR::create_node_df(ncol(adj_mat), label = node_labels)
  graph <- DiagrammeR::create_graph(nodes, directed = TRUE)
  adj_mat <- round(adj_mat, round_digit)
  for (i in 1:ncol(adj_mat)) {
    for (j in which(abs(adj_mat[, i]) > 0)) {
      graph <- DiagrammeR::add_edge(graph, from = i, to = j, edge_aes =
DiagrammeR::edge_aes(label = adj_mat[j, i]))
    }
  }
  dot <- DiagrammeR::render_graph(graph, layout = "dot")
  dot$x$diagram <- gsub("neato", "dot", dot$x$diagram)
  print(DiagrammeR::grViz(dot$x$diagram))
}

#---- Sample Data Generation -----

my_simulate_example = function(
  SampleSize = 200, Nonnormality = "None", Missingness = "None", Prop =
0.1){

  # Nonnormality : "None","Mild","Median","Severe"
  # Missingness : "None","MCAR","MAR","MNAR"
  Num_OV <- 6

  # Nonnormality
  if(tolower(Nonnormality) == "none"){
    mv_skewness <-  0.00;
    mv_kurtosis <- Num_OV*(Num_OV+2) + 0.00
  }
```

```
  if(tolower(Nonnormality) == "mild"){
    mv_skewness <-  0.76;
    mv_kurtosis <- Num_OV*(Num_OV+2) - 1.35
  }
  if(tolower(Nonnormality) == "median"){
    mv_skewness <-  3.08;
    mv_kurtosis <- Num_OV*(Num_OV+2) + 0.59
  }
  if(tolower(Nonnormality) == "severe"){
    mv_skewness <- 14.32;
    mv_kurtosis <- Num_OV*(Num_OV+2) + 7.47
  }

  # Psi
  vec_psi <- c(rep(0.50, Num_OV))
  Matrix_Psi <- diag(
    x = vec_psi, nrow = Num_OV, ncol = Num_OV)
  Psi <- mnonr::mnonr(
    n = SampleSize,
    p = Num_OV,
    ms = mv_skewness,
    mk = mv_kurtosis,
    Sigma = Matrix_Psi,
    initial = NULL)

  x4 <- Psi[, 4]
  x3 <- 6.00 * x4 + Psi[, 3]
  x1 <- 3.00 * x4 + Psi[, 1]
  x5 <- -1.00 * x3 + 8.00 * x1 + Psi[, 5]
  x2 <-  2.00 * x3 + 3.00 * x1 + Psi[, 2]
  x6 <-  4.00 * x1 + Psi[, 6]

  simdat <- as.data.frame(cbind(x1,x2,x3,x4,x5,x6))
  colnames(simdat) <- c("x1","x2","x3","x4","x5","x6")

  if (tolower(Missingness) != "none"){
    simdat <- mice::ampute(simdat, prop = Prop, mech =
toupper(Missingness))$amp}
  return(simdat)
}


MySimulate_PM1 <- function(
  SampleSize = 200, Nonnormality = "None", Missingness = "None", Prop =
0.1){

  # Out of latent confounding variable
  # Nonnormality : "None","Mild","Medium","High"
  # Missingness : "None","MCAR","MAR","MNAR"

  Num_OV <- 15;  Num_LV <- 3

  # Nonnormality
  if(tolower(Nonnormality) == "none"){
    skewness_Psi <-  0.00;
    skewness_Epsilon <- 0.00;
    kurtosis_Psi <- Num_LV*(Num_LV+2) + 0.00;
    kurtosis_Epsilon <- Num_OV*(Num_OV+2) + 0.00;
  }
  if(tolower(Nonnormality) == "mild"){
    skewness_Psi <-  1;
```

```
  skewness_Epsilon <- 1;
  kurtosis_Psi <- Num_LV*(Num_LV+2) + 32;
  kurtosis_Epsilon <- Num_OV*(Num_OV+2) + 32;
}
if(tolower(Nonnormality) == "medium"){
  skewness_Psi <-  3;
  skewness_Epsilon <- 3;
  kurtosis_Psi <- Num_LV*(Num_LV+2) + 61;
  kurtosis_Epsilon <- Num_OV*(Num_OV+2) + 61;
}
if(tolower(Nonnormality) == "severe"){
  skewness_Psi <-  15;
  skewness_Epsilon <- 15;
  kurtosis_Psi <- Num_LV*(Num_LV+2) + 91;
  kurtosis_Epsilon <- Num_OV*(Num_OV+2) + 91;
}

# Psi
vec_Psi <- c(0.64,1.00,0.64)
Matrix_Psi <- diag(vec_Psi, nrow = Num_LV, ncol = Num_LV)
Psi <- mnonr::mnonr(
  n = SampleSize,
  p = Num_LV,
  ms = skewness_Psi,
  mk = kurtosis_Psi,
  Sigma = Matrix_Psi,
  initial=NULL)

# Theta Epsilon
vec_thetaEpsilon <- c(
  .2895,.51,.51,.51,.2895,
  .51,.51,.51,.51,.51,
  .2895,.51,.51,.51,.51)
Matrix_ThetaEpsilon <- diag(
  x = vec_thetaEpsilon, nrow = Num_OV,ncol = Num_OV)
ThetaEpsilon <- mnonr::mnonr(
  n = SampleSize,
  p = Num_OV,
  ms = skewness_Epsilon,
  mk = kurtosis_Epsilon,
  Sigma = Matrix_ThetaEpsilon,
  initial=NULL)

# Lambda
Lambda <- data.frame(
  F1 = c(.70,.70,.70,.70,.70,.00,.00,.00,.00,.00,.00,.00,.00,.00),
  F2 = c(.21,.00,.00,.00,.00,.70,.70,.70,.70,.70,.00,.00,.00,.00,.00),
  F3 = c(.00,.00,.00,.00,.21,.00,.00,.00,.00,.21,.70,.70,.70,.70,.70))

Beta <- matrix(
  c(0.00,0.60,0.00,
    0.00,0.00,0.00,
    0.60,0.00,0.00),
  nrow = Num_LV, ncol = Num_LV, byrow = T)

F2 <- Psi[,2]
F1 <- Beta[1,2] * F2 + Psi[,1]
F3 <- Beta[3,1] * F1 + Psi[,3]

x1  <- Lambda[ 1,1] * F1 + ThetaEpsilon[, 1] + Lambda[ 1,2] * F2
x2  <- Lambda[ 2,1] * F1 + ThetaEpsilon[, 2]
```

```
  x3  <- Lambda[ 3,1] * F1 + ThetaEpsilon[, 3]
  x4  <- Lambda[ 4,1] * F1 + ThetaEpsilon[, 4]
  x5  <- Lambda[ 5,1] * F1 + ThetaEpsilon[, 5] + Lambda[ 5,3] * F3

  x6  <- Lambda[ 6,2] * F2 + ThetaEpsilon[, 6]
  x7  <- Lambda[ 7,2] * F2 + ThetaEpsilon[, 7]
  x8  <- Lambda[ 8,2] * F2 + ThetaEpsilon[, 8]
  x9  <- Lambda[ 9,2] * F2 + ThetaEpsilon[, 9]
  x10 <- Lambda[10,2] * F2 + ThetaEpsilon[,10] + Lambda[10,3] * F3

  x11 <- Lambda[11,3] * F3 + ThetaEpsilon[,11]
  x12 <- Lambda[12,3] * F3 + ThetaEpsilon[,12]
  x13 <- Lambda[13,3] * F3 + ThetaEpsilon[,13]
  x14 <- Lambda[14,3] * F3 + ThetaEpsilon[,14]
  x15 <- Lambda[15,3] * F3 + ThetaEpsilon[,15]

  simdat <- data.frame(
    cbind(x1,x2,x3,x4,x5,x6,x7,x8,
          x9,x10,x11,x12,x13,x14,x15))

  if (tolower(Missingness) != "none"){
    simdat <- mice::ampute(
      simdat, prop = Prop, mech = toupper(Missingness))$amp}
  return(simdat)
}


MySimulate_PM2 <- function(
  SampleSize = 200, Nonnormality = "None", Missingness = "None", Prop =
0.1){

  # Nonnormality : "None","Mild","Medium","Severe"
  # Missingness : "None","MCAR","MAR","MNAR"

  Num_OV <- 14;  Num_LV <- 4

  # Nonnormality
  if(tolower(Nonnormality) == "none"){
    skewness_Psi <-  0.00;
    skewness_Epsilon <- 0.00;
    kurtosis_Psi <- Num_LV*(Num_LV+2) + 0.00;
    kurtosis_Epsilon <- Num_OV*(Num_OV+2) + 0.00;
  }
  if(tolower(Nonnormality) == "mild"){
    skewness_Psi <-  1;
    skewness_Epsilon <- 1;
    kurtosis_Psi <- Num_LV*(Num_LV+2) + 32;
    kurtosis_Epsilon <- Num_OV*(Num_OV+2) + 32;
  }
  if(tolower(Nonnormality) == "medium"){
    skewness_Psi <-  3;
    skewness_Epsilon <- 3;
    kurtosis_Psi <- Num_LV*(Num_LV+2) + 61;
    kurtosis_Epsilon <- Num_OV*(Num_OV+2) + 61;
  }
  if(tolower(Nonnormality) == "severe"){
    skewness_Psi <-  15;
    skewness_Epsilon <- 15;
    kurtosis_Psi <- Num_LV*(Num_LV+2) + 91;
    kurtosis_Epsilon <- Num_OV*(Num_OV+2) + 91;
  }
```

```
# Psi
vec_Psi <- c(1.00,1.00,0.64,0.64)
Matrix_Psi <- diag(vec_Psi, nrow = Num_LV, ncol = Num_LV)
Psi <- mnonr::mnonr(
  n = SampleSize,
  p = Num_LV,
  ms = skewness_Psi,
  mk = kurtosis_Psi,
  Sigma = Matrix_Psi,
  initial=NULL)

# Theta Epsilon
vec_thetaEpsilon <- c(
  .40,.37,.32,
  .51,.56,.54,
  .66,.41,.48,
  .56,.27,.75,.63,.56)
Matrix_ThetaEpsilon <- diag(
  x = vec_thetaEpsilon, nrow = Num_OV, ncol = Num_OV)
ThetaEpsilon <- mnonr::mnonr(
  n = SampleSize,
  p = Num_OV,
  ms = skewness_Epsilon,
  mk = kurtosis_Epsilon,
  Sigma = Matrix_ThetaEpsilon,
  initial=NULL)

# Lambda
Lambda <- data.frame(
  AVO = c(.77,.80,.82,rep(.00,11)),
  ANX = c(rep(.00,3),.70,.66,.68,rep(.00,8)),
  COP = c(rep(.00,6),.58,.77,.72,rep(.00,5)),
  DIS = c(rep(.00,9),.66,.86,.50,.61,.66))

Beta <- matrix(
  c(0.00,0.00,0.00,0.00,
    0.00,0.00,0.00,0.00,
    0.25,0.50,0.00,0.00,
    0.22,0.00,0.76,0.00),
  nrow = Num_LV, ncol = Num_LV, byrow = T)

AVO <- Psi[,1]
ANX <- Psi[,2]
COP <- Beta[3,1] * AVO + Beta[3,2] * ANX + Psi[,3]
DIS <- Beta[4,3] * COP + Beta[4,1] * AVO + Psi[,4]

x1  <- Lambda[ 1,1] * AVO + ThetaEpsilon[, 1]
x2  <- Lambda[ 2,1] * AVO + ThetaEpsilon[, 2]
x3  <- Lambda[ 3,1] * AVO + ThetaEpsilon[, 3]

x4  <- Lambda[ 4,2] * ANX + ThetaEpsilon[, 4]
x5  <- Lambda[ 5,2] * ANX + ThetaEpsilon[, 5]
x6  <- Lambda[ 6,2] * ANX + ThetaEpsilon[, 6]

x7  <- Lambda[ 7,3] * COP + ThetaEpsilon[, 7]
x8  <- Lambda[ 8,3] * COP + ThetaEpsilon[, 8]
x9  <- Lambda[ 9,3] * COP + ThetaEpsilon[, 9]

x10 <- Lambda[10,4] * DIS + ThetaEpsilon[,10]
x11 <- Lambda[11,4] * DIS + ThetaEpsilon[,11]
```

```
  x12 <- Lambda[12,4] * DIS + ThetaEpsilon[,12]
  x13 <- Lambda[13,4] * DIS + ThetaEpsilon[,13]
  x14 <- Lambda[14,4] * DIS + ThetaEpsilon[,14]

  simdat <- data.frame(
    cbind(x1,x2,x3,x4,x5,x6,x7,x8,
          x9,x10,x11,x12,x13,x14))

  if (tolower(Missingness) != "none"){
    simdat <- mice::ampute(simdat, prop = Prop, mech =
toupper(Missingness))$amp}
  return(simdat)
}


MySimulate_PM3 <- function(
  SampleSize = 200, Nonnormality = "None", Missingness = "None", Prop =
0.1){

  # Nonnormality : "None","Mild","Medium","Severe"
  # Missingness : "None","MCAR","MAR","MNAR"

  Num_OV <- 24;  Num_LV <- 6

  # Nonnormality
  if(tolower(Nonnormality) == "none"){
    skewness_Psi <-  0.00;
    skewness_Epsilon <- 0.00;
    kurtosis_Psi <- Num_LV*(Num_LV+2) + 0.00;
    kurtosis_Epsilon <- Num_OV*(Num_OV+2) + 0.00;
  }
  if(tolower(Nonnormality) == "mild"){
    skewness_Psi <-  1;
    skewness_Epsilon <- 1;
    kurtosis_Psi <- Num_LV*(Num_LV+2) + 32;
    kurtosis_Epsilon <- Num_OV*(Num_OV+2) + 32;
  }
  if(tolower(Nonnormality) == "medium"){
    skewness_Psi <-  3;
    skewness_Epsilon <- 3;
    kurtosis_Psi <- Num_LV*(Num_LV+2) + 61;
    kurtosis_Epsilon <- Num_OV*(Num_OV+2) + 61;
  }
  if(tolower(Nonnormality) == "severe"){
    skewness_Psi <-  15;
    skewness_Epsilon <- 15;
    kurtosis_Psi <- Num_LV*(Num_LV+2) + 91;
    kurtosis_Epsilon <- Num_OV*(Num_OV+2) + 91;
  }

  # Psi
  vec_Psi <- c(1.00,0.64,0.64,1.00,0.64,0.64)
  Matrix_Psi <- diag(vec_Psi, nrow = Num_LV, ncol = Num_LV)
  Psi <- mnonr::mnonr(
    n = SampleSize,
    p = Num_LV,
    ms = skewness_Psi,
    mk = kurtosis_Psi,
    Sigma = Matrix_Psi,
    initial=NULL)
```

```
# Theta Epsilon
residual_list <- c(.510,.510,.510,.510)
vec_thetaEpsilon <- c(rep(residual_list,Num_LV))
Matrix_ThetaEpsilon <- diag(
  x = vec_thetaEpsilon, nrow = Num_OV,ncol = Num_OV)

Matrix_ThetaEpsilon[ 1, 5] <- Matrix_ThetaEpsilon[ 5, 1] <- .10
Matrix_ThetaEpsilon[ 5, 9] <- Matrix_ThetaEpsilon[ 9, 5] <- .10
Matrix_ThetaEpsilon[ 1, 9] <- Matrix_ThetaEpsilon[ 9, 1] <- .10

Matrix_ThetaEpsilon[ 2, 6] <- Matrix_ThetaEpsilon[ 6, 2] <- .10
Matrix_ThetaEpsilon[ 6,10] <- Matrix_ThetaEpsilon[10, 6] <- .10
Matrix_ThetaEpsilon[ 2,10] <- Matrix_ThetaEpsilon[10, 2] <- .10

Matrix_ThetaEpsilon[ 3, 7] <- Matrix_ThetaEpsilon[ 7, 3] <- .10
Matrix_ThetaEpsilon[ 7,11] <- Matrix_ThetaEpsilon[11, 7] <- .10
Matrix_ThetaEpsilon[ 3,11] <- Matrix_ThetaEpsilon[11, 3] <- .10

Matrix_ThetaEpsilon[ 4, 8] <- Matrix_ThetaEpsilon[ 8, 4] <- .10
Matrix_ThetaEpsilon[ 8,12] <- Matrix_ThetaEpsilon[12, 8] <- .10
Matrix_ThetaEpsilon[ 4,12] <- Matrix_ThetaEpsilon[12, 4] <- .10

Matrix_ThetaEpsilon[13,17] <- Matrix_ThetaEpsilon[17,13] <- .10
Matrix_ThetaEpsilon[17,21] <- Matrix_ThetaEpsilon[21,17] <- .10
Matrix_ThetaEpsilon[13,21] <- Matrix_ThetaEpsilon[21,13] <- .10

Matrix_ThetaEpsilon[14,18] <- Matrix_ThetaEpsilon[18,14] <- .10
Matrix_ThetaEpsilon[18,22] <- Matrix_ThetaEpsilon[22,18] <- .10
Matrix_ThetaEpsilon[14,22] <- Matrix_ThetaEpsilon[22,14] <- .10

Matrix_ThetaEpsilon[15,19] <- Matrix_ThetaEpsilon[19,15] <- .10
Matrix_ThetaEpsilon[19,23] <- Matrix_ThetaEpsilon[23,19] <- .10
Matrix_ThetaEpsilon[15,23] <- Matrix_ThetaEpsilon[23,15] <- .10

Matrix_ThetaEpsilon[16,20] <- Matrix_ThetaEpsilon[20,16] <- .10
Matrix_ThetaEpsilon[20,24] <- Matrix_ThetaEpsilon[24,20] <- .10
Matrix_ThetaEpsilon[16,24] <- Matrix_ThetaEpsilon[24,16] <- .10

ThetaEpsilon <- mnonr::mnonr(
  n = SampleSize,
  p = Num_OV,
  ms = skewness_Epsilon,
  mk = kurtosis_Epsilon,
  Sigma = Matrix_ThetaEpsilon,
  initial=NULL)

# Lambda
Lambda <- data.frame(
  F11 = c(c(.70,.70,.70,.70),rep(.00, 4*5)),
  F12 = c(rep(.00, 4*1),c(.70,.70,.70,.70),rep(.00, 4*4)),
  F13 = c(rep(.00, 4*2),c(.70,.70,.70,.70),rep(.00, 4*3)),
  F21 = c(rep(.00, 4*3),c(.70,.70,.70,.70),rep(.00, 4*2)),
  F22 = c(rep(.00, 4*4),c(.70,.70,.70,.70),rep(.00, 4*1)),
  F23 = c(rep(.00, 4*5),c(.70,.70,.70,.70)))

Beta <- matrix(
  c(0.00,0.00,0.00,0.00,0.00,0.00,
    0.60,0.00,0.00,0.40,0.00,0.00,
    0.00,0.60,0.00,0.00,0.40,0.00,
    0.00,0.00,0.00,0.00,0.00,0.00,
    0.00,0.00,0.00,0.60,0.00,0.00,
```

```
    0.00,0.00,0.00,0.00,0.60,0.00),
  nrow = Num_LV, ncol = Num_LV, byrow = T)

F11 <- Psi[,1];  F21 <- Psi[,4]
F12 <- Beta[2,1] * F11 + Beta[2,4] * F21 + Psi[,2]
F22 <- Beta[5,4] * F21 + Psi[,5]
F13 <- Beta[3,2] * F12 + Beta[3,5] * F22 + Psi[,3]
F23 <- Beta[6,5] * F22 + Psi[,6]

x1  <- Lambda[ 1,1] * F11 + ThetaEpsilon[, 1]
x2  <- Lambda[ 2,1] * F11 + ThetaEpsilon[, 2]
x3  <- Lambda[ 3,1] * F11 + ThetaEpsilon[, 3]
x4  <- Lambda[ 4,1] * F11 + ThetaEpsilon[, 4]

x5  <- Lambda[ 5,2] * F12 + ThetaEpsilon[, 5]
x6  <- Lambda[ 6,2] * F12 + ThetaEpsilon[, 6]
x7  <- Lambda[ 7,2] * F12 + ThetaEpsilon[, 7]
x8  <- Lambda[ 8,2] * F12 + ThetaEpsilon[, 8]

x9  <- Lambda[ 9,3] * F13 + ThetaEpsilon[, 9]
x10 <- Lambda[10,3] * F13 + ThetaEpsilon[,10]
x11 <- Lambda[11,3] * F13 + ThetaEpsilon[,11]
x12 <- Lambda[12,3] * F13 + ThetaEpsilon[,12]

x13 <- Lambda[13,4] * F21 + ThetaEpsilon[,13]
x14 <- Lambda[14,4] * F21 + ThetaEpsilon[,14]
x15 <- Lambda[15,4] * F21 + ThetaEpsilon[,15]
x16 <- Lambda[16,4] * F21 + ThetaEpsilon[,16]

x17 <- Lambda[17,5] * F22 + ThetaEpsilon[,17]
x18 <- Lambda[18,5] * F22 + ThetaEpsilon[,18]
x19 <- Lambda[19,5] * F22 + ThetaEpsilon[,19]
x20 <- Lambda[20,5] * F22 + ThetaEpsilon[,20]

x21 <- Lambda[21,6] * F23 + ThetaEpsilon[,21]
x22 <- Lambda[22,6] * F23 + ThetaEpsilon[,22]
x23 <- Lambda[23,6] * F23 + ThetaEpsilon[,23]
x24 <- Lambda[24,6] * F23 + ThetaEpsilon[,24]

simdat <- data.frame(
  cbind(x1,x2,x3,x4,x5,x6,x7,x8,
        x9,x10,x11,x12,x13,x14,x15,x16,
        x17,x18,x19,x20,x21,x22,x23,x24))

if (tolower(Missingness) != "none"){
  simdat <- mice::ampute(
    simdat, prop = Prop, mech = toupper(Missingness))$amp}
return(simdat)
}
```