2018

EXAMINING THE CONFIRMATORY TETRAD ANALYSIS (CTA) AS A SOLUTION OF THE INADEQUACY OF TRADITIONAL STRUCTURAL EQUATION MODELING (SEM) FIT INDICES

Hangcheng Liu

EXAMINING THE CONFIRMATORY TETRAD ANALYSIS (CTA) AS A SOLUTION OF THE INADEQUACY OF TRADITIONAL STRUCTURAL EQUATION MODELING (SEM) FIT INDICES

A dissertation submitted in partial fulfillment of the requirements for the Doctor of Philosophy in Biostatistics at Virginia Commonwealth University

By

Hangcheng Liu

Doctor of Philosophy

Director: Dr. Robert A. Perera

Assistant Professor

Department of Biostatistics

Virginia Commonwealth University

Richmond, Virginia

August, 2018

Acknowledgment

I would like to thank my advisor Dr. Perera for the continuous support of my dissertation and related research, for his patience, motivation, and immense knowledge. The door to Dr. Perera office was always open whenever I ran into a trouble spot or had a question about my research or writing. His guidance helped me in all the time of research and writing of this dissertation. I could not have imagined having a better advisor for my Ph.D. study.

I also would like to thank my dissertation committee members Dr. Riddle, Dr. Kang, Dr. Chen, and Dr. Dahman for their advice with this research. Also, I am grateful to the Department of Biostatistics at Virginia Commonwealth University for offering me this valuable opportunity of study.

Last but not the least, I would like to thank my family: my parents, my brother and my wife Aobo for supporting me spiritually throughout writing this dissertation and my life in general.

TABLE OF CONTENTS

List of Tables

List of Figures

Abstract

Structural Equation Modeling (SEM) is a framework of statistical methods that allows us to represent complex relationships between variables. SEM is widely used in economics, genetics and the behavioral sciences (e.g. psychology, psychobiology, sociology and medicine). Model complexity is defined as a model's ability to fit different data patterns and it plays an important role in model selection when applying SEM. As in linear regression, the number of free model parameters is typically used in traditional SEM model fit indices as a measure of the model complexity. However, only using number of free model parameters to indicate SEM model complexity is crude since other contributing factors, such as the type of constraint or functional form are ignored.

To solve this problem, a special technique, Confirmatory Tetrad Analysis (CTA) is examined. A *tetrad* refers to the difference in the products of certain covariances (or correlations) among four random variables. A structural equation model often implies that some tetrads should be zero. These model implied zero tetrads are called *vanishing tetrads*. In CTA, the goodness of fit can be determined by testing the null hypothesis that the model implied vanishing tetrads are equal to zero. CTA can be helpful to improve model selection because different functional forms may affect the model implied vanishing tetrad number (t), and models not nested according to the traditional likelihood ratio test may be nested in terms of tetrads.

In this dissertation, an R package was created to perform CTA, a two-step method was developed to determine SEM model complexity using simulated data, and it is demonstrated how the number of vanishing tetrads can be helpful to indicate SEM model complexity in some situations.

KEY WORDS:
Structural Equation Modeling; Model complexity; Confirmatory Tetrad Analysis; Model selection; R package; Simulated data

Chapter 1 Introduction

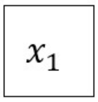
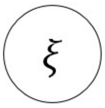
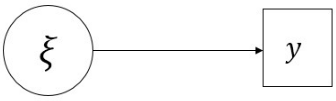
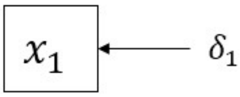
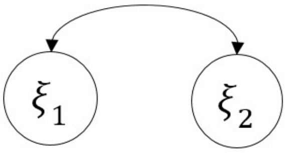
Structural Equation Modeling

Structural Equation Modeling (SEM) is a series of statistical methods that allow for the statistical modeling of complex relationships between one or more independent variables and one or more dependent variables. SEM is widely used in the behavioral sciences (e.g. psychology, psychobiology, sociology). The history of SEM traces back to three different traditions: (1) factor analysis from Charles Spearman (Spearman, 1904), (2) path analysis and tracing rules developed by the geneticist Sewall Wright (Wright, 1934), (3) simultaneous-equation models, as developed in economics. In the early 1970s, many different researchers made significant contributions to merge these three traditions (Bollen 2014). After the first SEM software LISREL (LISREL represents for Linear Structural RELations) developed by Karl Jöreskog and Dag Sörbom in 1973, was introduced, more researchers accessed this new statistical approach of SEM and many advances were made (Rosseel, 2012).

Path analysis, first described by Sewall Wright can be seen as a precursor of SEM. It provides three important tools that contribute to building a SEM model: (1) path diagram, (2) decomposition of covariances and correlations variables, (3) decomposition of effects in a specific model.

Generally, drawing a path diagram is always the first step to build a SEM model, it can be seen as a pictorial representation of a system of simultaneous equations. For many researchers this picture may represent the relationships more clearly than the equations (Bollen 2014). Symbols used in path diagrams are listed in Table 1.1.1.

Table 1.1.1 Symbols in Path diagrams

	A box represents an observed variable
	A circle or ellipse represents an unobserved or latent variable
	A single headed straight arrow represents the influence ("cause") of one variable on another
	A unenclosed variable represents an error term
	A double-headed curved arrow represents a covariance or correlation between two variables

A widely used general representation of structural equations with observed variables from Bollen (1989) is shown below:

$$\boldsymbol{\eta} = \mathbf{B}\boldsymbol{\eta} + \boldsymbol{\Gamma}\boldsymbol{\xi} + \boldsymbol{\zeta} \tag{1.1.1}$$

$$\mathbf{y} = \boldsymbol{\Lambda}_y\boldsymbol{\eta} + \boldsymbol{\varepsilon} \tag{1.1.2}$$

$$\mathbf{x} = \boldsymbol{\Lambda}_x\boldsymbol{\xi} + \boldsymbol{\delta} \tag{1.1.3}$$

Function (1.1.1) is called the latent variable model.

where $\boldsymbol{\eta}$ = $m\times 1$ vector of latent endogenous random variables

$\boldsymbol{\xi}$ = $n\times 1$ vector of latent exogenous random variables

$\mathbf{B}$ = $m\times m$ coefficient matrix (called "beta")

$\boldsymbol{\Gamma}$ = $m\times n$ coefficient matrix (called "gamma")

$\boldsymbol{\zeta}$ = $p\times 1$ vector of errors (called "zeta"), $E(\zeta) = 0$, uncorrelated with ξ

Functions (1.1.2) and (1.1.3) are called the measurement model.

where $\mathbf{y}$ = $p\times 1$ vector of observed endogenous variables

$\mathbf{x}$ = $q\times 1$ vector of observed exogenous variables

$\boldsymbol{\Lambda}_y$ = $p\times n$ coefficient matrix

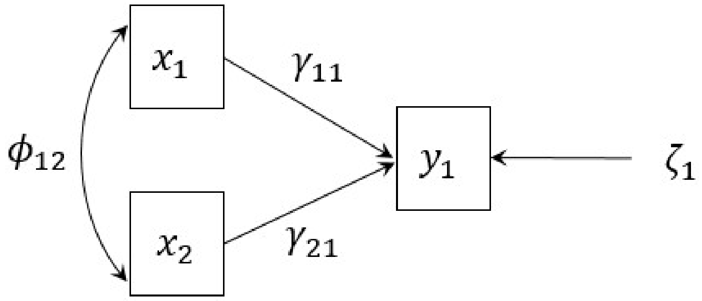
$\boldsymbol{\Lambda}_x$ = $q\times n$ coefficient matrix

$\boldsymbol{\varepsilon}$ = $p\times 1$ the errors of measurement for $\mathbf{y}$, $E(\boldsymbol{\varepsilon}) = 0$, uncorrelated with $\boldsymbol{\delta}$

$\boldsymbol{\delta}$ = $q\times 1$ the errors of measurement for $\mathbf{x}$, $E(\boldsymbol{\delta}) = 0$, uncorrelated with $\boldsymbol{\varepsilon}$

A simple example of path diagram is shown in Figure 1.1.1 and represents a multiple regression model with two predictors.

Figure 1.1.1 Path diagram of a simple SEM model



There is no measurement error or latent variables in this example, thus Equation 1.1.2 and Equation 1.1.3 are not necessary, and Equation 1.1.1 becomes

$$\mathbf{y} = \mathbf{B}\mathbf{y} + \mathbf{\Gamma}\mathbf{x} + \boldsymbol{\zeta} \tag{1.1.4}$$

where $\mathbf{y}$ = p×1 vector of observed endogenous variables

$\mathbf{x}$ = q×1 vector of observed exogenous variables

$\mathbf{B}$ = p×p coefficient matrix (called "beta")

$\mathbf{\Gamma}$ = p×q coefficient matrix (called "gamma")

$\boldsymbol{\zeta}$ = p×1 vector of errors (called "zeta")

$\mathbf{\Psi}$ = p×p covariance matrix of errors (called "psi")

$\mathbf{\Phi}$ = q×q covariance matrix of exogenous variables (called "phi")

In SEM, a variable that is not causally related to another variable in the model is called an exogenous variable, like x_1 and x_2 in Figure 1.1.1. A variable that is causally related to one or more variables in the model is called endogenous variable, like y_1 in Figure 1.1.1. In this example, number of endogenous variables is 1(p=1) and number of exogenous variables is 2 (q=2).

Apply Equation 1.1.4 to the path diagram in Figure 1.1.1, the results are:

$$[y_1] = [0] * [y_1] + [\gamma_{11} \quad \gamma_{21}] * \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + [\zeta_1] \qquad (1.1.5)$$

$$\boldsymbol{\Psi} = [\psi_{11}] \qquad \boldsymbol{\Phi} = \begin{bmatrix} \phi_{11} & \phi_{12} \\ \phi_{21} & \phi_{22} \end{bmatrix}$$

After the theorized models are specified, the next step is to check the identification of these models, which means determining if a unique solution exists for estimates of $\boldsymbol{\Lambda}$, $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$. One necessary but not sufficient rule is the number of parameters to be estimated (free parameters implied in the model) must be less than the number of known parameters. This is also an important role as it defines the degrees of freedom in a SEM model and requires that they be greater than zero. Known parameters are the non-redundant items in the observed covariance matrix. Use p to represent the number of observed y variables and q to represent observed x variables, the known parameter number is:

$$\frac{(p+q)(p+q+1)}{2}$$

Use t to represent the number of free parameters in a model (t is the total number of λ's, γ's, ϕ's, and ψ's). An identified SEM model must have $\frac{1}{2}(p+q)(p+q+1) > t$.

And the degrees of freedom of this SEM model can be calculated by:

$$df = 1/2\,(p+q)(p+q+1) - t.$$

After the identification, the next step is to estimate the parameter values in these specified models. SEM is commonly conceptualized as a hybrid between some form of analysis of variance (ANOVA)/regression and some form of factor analysis (Bollen, 2014). However, there is a difference between SEM and multiple regression/ANOVA in the concept of estimating the model. In multiple regression/ANOVA, the regression coefficients derive from the minimization of the sum of squared errors between the predicted and observed dependent variable, or the coefficients are derived from the maximization of the likelihood of observed raw data. However, SEM minimizes the difference between the sample covariances and the covariances predicted by the model. If the model were correct and if all parameters of the model were known, the population covariance matrix would be exactly reproduced. The fundamental hypothesis in SEM can be shown as:

$$\Sigma = \Sigma(\theta) \tag{1.1.6}$$

In Equation (1.1.6), Σ is the population covariance matrix of observed variables, θ is a vector that contains the model parameters, and $\Sigma(\theta)$ is the covariance matrix written as a function of θ. This equation provides a unified way of including many of the most widely used statistical techniques. Regression analysis, simultaneous equation systems, confirmatory factor analysis, canonical correlations, panel data analysis, ANOVA, analysis of covariance, and multiple indicator models all can be seen as special cases of (1.1.1, 1.1.2 &1.1.3).

Appling some linear algebra to Equation 1.1.4, the model implied covariance matrix can be found by calculating the expectations of $\mathbf{xx}'$, $\mathbf{yy}'$, and $\mathbf{xy}'$:

$$\Sigma_{yy}(\theta) = \mathrm{E}(\mathbf{yy}') = \mathrm{E}[(\mathbf{By} + \boldsymbol{\Gamma}\mathbf{x} + \boldsymbol{\zeta}) * (\mathbf{By} + \boldsymbol{\Gamma}\mathbf{x} + \boldsymbol{\zeta})']$$

$$= (\mathbf{I} - \mathbf{B})^{-1}(\boldsymbol{\Gamma\Phi\Gamma}' + \boldsymbol{\Psi})(\mathbf{I} - \mathbf{B})^{-1\prime}$$

$$\Sigma_{xy}(\theta) = \mathrm{E}(\mathbf{xy}') = \mathrm{E}[\mathbf{x} * (\mathbf{By} + \boldsymbol{\Gamma}\mathbf{x} + \boldsymbol{\zeta})] = \boldsymbol{\Phi\Gamma}' * (\mathbf{I} - \mathbf{B})^{-1\prime}$$

$$\Sigma_{xx}(\theta) = \mathrm{E}(\mathbf{xx}') = \boldsymbol{\Phi} \tag{1.1.7}$$

where **I** is an identity matrix of size p (p=number of endogenous variables).

Then the model implied covariance matrix $\Sigma(\theta)$ can be shown as:

$$\Sigma(\theta) = \begin{bmatrix} \Sigma_{yy}(\theta) & \Sigma_{xy}(\theta) \\ \Sigma_{xy}(\theta) & \Sigma_{xx}(\theta) \end{bmatrix}$$

$$= \begin{bmatrix} (\mathbf{I} - \mathbf{B})^{-1}(\boldsymbol{\Gamma\Phi\Gamma}' + \boldsymbol{\Psi})(\mathbf{I} - \mathbf{B})^{-1\prime} & (\mathbf{I} - \mathbf{B})^{-1}\boldsymbol{\Gamma\Phi} \\ \boldsymbol{\Phi\Gamma}'(\mathbf{I} - \mathbf{B})^{-1\prime} & \boldsymbol{\Phi} \end{bmatrix} \tag{1.1.8}$$

To estimate the parameters in the equation $\mathbf{y} = \mathbf{By} + \boldsymbol{\Gamma}\mathbf{x} + \boldsymbol{\zeta}$, structural equation models encompass a wide range of multivariate statistical techniques. Bollen (2014) identified three components in today's general structural equation models: (1) path analysis, (2) the conceptual synthesis of latent variable and measurement models, and (3) general estimation procedures. The confirmatory tetrad analysis (CTA) will be discussed later and is one of the newest methods for building structural equation models.

1.2 Fit Indices in the Selection of Structural Equation Models

Testing the model fit for all selected models that best represents the data (the sample covariance or correlation matrix) is a crucial part of structural equation modeling, after the potential structural equations models were selected and high quality data were selected. The population covariance is noted as Σ and the covariance produced by the proposed model is noted as $\Sigma(\theta)$, where θ represents the parameters in the model. Thus the null hypothesis for the global model fit test in SEM is $\mathrm{H}_0\colon \Sigma = \Sigma(\theta)$. For a particular sample dataset, the sample covariance matrix S can be measured and the potential structural equations based on S can be marked as $\Sigma\left(\hat{\theta}\right)$. Thus the global fit can be measured by comparing the sample covariance matrix S and the implied covariance matrix $\Sigma\left(\hat{\theta}\right)$ that determined by the estimate of θ. Ideally, if $\mathrm{H}_0\colon \Sigma = \Sigma(\theta)$ is true, the population residual covariance matrix $\Sigma(\mathrm{e}) = \Sigma - \Sigma(\theta)$ should be a zero matrix. However, this situation will not happen in most cases. In the residual covariance matrix, the element for ith row and jth column can be marked as e_{ij}, a positive value for e_{ij} means that the structural equations under estimated the covariance between variable i and j, while a negative value means the predicted covariance is too high. Once the sample covariance matrix (S) and estimated covariance implied by the model ($\Sigma\left(\hat{\theta}\right)$) is obtained, fitting functions are used to test the overall discrepancy between S and $\Sigma\left(\hat{\theta}\right)$.

In sum, the sample residual matrix $S - \Sigma(\hat{\theta})$ can identify the sample variance or covariance elements in a SEM model, and it would be helpful to have tests of whether the residuals depart from the population values of zero. The difference between the population covariance/correlation matrix (Σ) and sample covariance/correlation matrix (S) is affected by sample size *N*, such that when *N* gets larger, the differences between the sample and the population become smaller. Tests of fit, therefore should consider sample size.

To test the overall model fit in SEM, the maximum likelihood fitting function developed by Jöreskog (1967) is one of the most popular tests of fit. It offers an inference about the relationship between estimated covariance/correlation matrix and the sample covariance/correlation matrix, it can be shown as:

$$F_{ML} = \log\left|\Sigma(\hat{\theta})\right| + tr\left(S * \Sigma(\hat{\theta})^{-1}\right) - \log|S| - (p + q)$$

where "$|\cdot|$" Indicates the determinant of a matrix, "tr" indicates the trace and $p + q$ is the total number of manifest variables in the model (p is the number of $\mathbf{y}$ in the model, and q is the number of $\mathbf{x}$ in the model).

When $\Sigma(\hat{\theta}) = S$, F_{ML} equals to zero:

$$F_{ML} = \log|S| + tr(S * S^{-1}) - \log|S| - (p + q) = tr(I) - (p + q) = 0$$

Thus, if a model can predict the values of the sample covariance matrix perfectly, it's F_{ML} should equal zero.

The assumption of this maximum likelihood fitting function is multivariate normality of the variables in the model. If the assumption is met, the relationship between S and $\Sigma(\hat{\theta})$ can be determined by the value of F_{ML}.

Using the maximum likelihood fitting functions, the validity of estimated models can be tested, and there are many fit indices which use the maximum likelihood function.

One example is the chi-square test, based on the fact that if the null hypothesis is true ($\mathrm{H}_0\colon \Sigma = \Sigma(\theta)$), $(N-1)$*F_{ML} will follow an asymptotic chi-square distribution (Bollen, 2014).

Under the null hypothesis that $\Sigma = \Sigma(\theta)$, the specification of the fixed, free and constrained parameters is valid. Form $\Sigma(\hat{\theta})$ as the sample predicted covariance matrix of the observed variables under H_0, the log likelihood is:

$$log\ L_0 = -\frac{(N-1)}{2}\{\log\left|\Sigma(\hat{\theta})\right| + tr\left(S * \Sigma(\hat{\theta})^{-1}\right)\}$$

Under the alternative hypothesis, set $\Sigma(\hat{\theta}) = S$ can maximize the log likelihood:

$$log\ L_1 = -\frac{(N-1)}{2}\{\log|S| + tr(S * S^{-1})\} = -\frac{(N-1)}{2}\{\log|S| + p + q\}$$

And $log(L_0\ /L_1)$ multiplied by -2 will follow the chi-square distribution when sample size *N* is large. Its degrees of freedom are $\frac{1}{2}(p+q)(p+q+1) - t$, where t is the number of free parameters in θ (t is the total number of λ's, ϕ's, and ψ's in the model):

$$-2\ log\left(\frac{L_0}{L_1}\right) = (N-1) * F_{ML}$$

Thus, when comparing $(N-1) * F_{ML}$ to the critical values of chi-square distribution, the overall model fit can be determined.

Generally, overall model fit tests are affected by four factors: The first factor is the departure of Σ from $\Sigma(\theta)$, which is what we want to determine by using the model fit tests. The second factor is the scales of the observed variables, for instance, if observed variables are measured in different units, the magnitudes of the variables will vary and in this situation, the sample covariance matrix should not be used to determine model fit (i.e. one would use the sample correlation matrix instead). Third, the sample size both affects the sample covariance matrix and the sample correlation matrix, in general cases, S converges to Σ and $\Sigma(\hat{\theta})$ converges to $\Sigma(\theta)$ as sample size increases. The fourth factor is random sampling variability. If the sample data do not meet the assumptions of statistical tests, the validity of the test result will be questioned. Therefore, in the interest to detect the departure of Σ from $\Sigma(\theta)$, effects from the scales of the observed variables r and the effect from sample size should be minimized.

In an attempt to improve the measurement of model fit in different situations, statisticians have sought and developed several criteria that reflect many facets of model fit. These criteria are also called fit indices. They all have their own strengths and limitations and should be used in the most appropriate situation. For example, the F_{ML} mentioned in the maximum likelihood fitting function can be counted as a fit index, when chi-square test is used, the limitations of chi-square test (such as sensitive to sample size) follow. Many statisticians have attempted to overcome these limitations, and tried to find the better ways to reflect model fit. As a result, a variety of fit indices are available. The most widely respected and reported fit indicates can be classified as (1) absolute fit indices, (2) incremental fit indices, (3) parsimony fit indices.

1.2.1 Absolute Fit Indices

Absolute fit indices are used to determine how well a model fits the sample data and are used to demonstrate which model has the best fit among several proposed models. These measures provide the most fundamental indication of how well the proposed models fit the data. Their calculations rely on measures of how well the model fits in comparison to no model at all (Jöreskog and Sörbom, 1993). Included in this category are the Chi-Squared test, Root Mean Square Error of Approximation (RMSEA), Goodness of Fit (GFI), Adjusted Goodness of Fit (AGFI), the Root Mean Square Residual (RMR) and the Standardized Root Mean Square Residual (SRMR). Their recommended threshold levels are listed in table 1.2.1.

Table 1.2.1 Absolute fit indices and their acceptable thresholds(Moss, 2009)

Fit Index	**Acceptable Threshold Levels**	**Description**
Chi-Square χ^2	p-value > 0.05	Hypothesis test of $\Sigma = \Sigma(\theta)$
RMSEA	Values less than 0.05 or 0.07	Has a known distribution. Favours parsimony. Values less than 0.03 represent excellent fit.
GFI	Values greater than 0.95	Scaled between 0 and 1, with higher values indicating better model fit.
AGFI	Values greater than 0.95	Adjusts the GFI based on the number of parameters in the model. Values can fall outside the 0-1.0 range.
RMR	Good models have small RMR	Residual based, unstandardized, the average squared differences between the residuals of the sample covariances and the residuals of the estimated covariances.
SRMR	Values less than 0.08	Standardized version of the RMR. Easier to interpret due to its standardized nature.

Among the absolute fit indices, Chi-square (χ^2) is the original fit index for structural models and is derived from the maximum likelihood fitting function ($\chi_M^2 = (N-1) * F_{ML}$, where N is the total sample size), it is also a direct test of the null hypothesis, so it is routinely reported in all SEM results sections. However, it is not considered to be a very useful fit index by most researchers, because it is affected by many factors, i.e. sample size, variable numbers, and distribution of variables. All absolute indices with the exception of the SRMR have similar problems to those of the chi-square, because they are transformations of the chi-square. To minimize these problems, alternative measures of chi-square fit have been developed. For example, the chi-square test is generally a reasonable fit index when the sample size is between 75 and 200 (Kenny, 2015). For sample sizes beyond this range, chi-square test tends to reject the null hypothesis for large samples and lacks power for small samples. One alternative fit measure is the Relative Chi-square. It uses the ratio of chi-square value and the degrees of freedom (*df*) as a fit index for the model tominimize the impact of sample size on the chi-square test result.

The RMSEA was first developed by Steiger and Lind (1980, cited in Steiger, 1990) and currently it's a popular measure of absolute model fit, its formula is shown below:

$$\text{RMSEA} = \hat{\varepsilon} = \frac{\sqrt{\hat{\Delta}_M}}{\sqrt{df_M * (N-1)}}$$

$$\hat{\Delta}_M = \max(0, \chi_M^2 - df_M)$$

where N is the sample size and df is the degrees of freedom of the model.

A universal threshold of RMSEA≤0.05 (or 0.07) is often used to indicate a good fit. This is a one sided test and the null hypothesis is $\varepsilon_0 \leq 0.05$. The RMSEA is usually reported in computer output with the 90% CI $[\hat{\epsilon}_L, \hat{\epsilon}_U]$. $\hat{\epsilon}_U \leq 0.05$ indicates close fit and $\hat{\epsilon}_L \geq 0.05$ indicates a not close fit. When the 90% CI contains 0.05 (or 0.07), the probability $P(\varepsilon_0 \leq 0.05)$ should be calculated and compared to the significant level like $\alpha = 0.05$ to make the judgement. RMSEA is largely based on the Relative Chi-square, thus it is also sensitive to the number of estimated parameters in the model. In other words, it tends to choose the model with less parameters. An advantage of RMSEA is that a confidence interval can be calculated for the RMSEA, which providesinformation regarding the precision RMSEA estimate.

The GFI is another transformation of the chi-square and it relies on the proportion of variance and covariances in the proposed model, and therefore determines the model's ability of reproduce the observed covariance matrix. The AGFI adjusts the GFI on degrees of freedom, like RMSEA, it penalizes complicated models.

The RMR and the SRMR are defined as the square root of the difference between the residuals of the sample covariance (correlation) matrix and the model implied covariance (correlation) matrix and they do not penalize for model complexity. Calculation functions for RMR and SRMR are:

$$\mathrm{RMR} = \sqrt{\sum_{i=1}^{q} \sum_{j=1}^{i} \frac{(s_{ij} - \sigma(\hat{\theta})_{ij})^2}{q(q+1)}}$$

$$\mathrm{SRMR} = \sqrt{\sum_{i=1}^{q} \sum_{j=1}^{i} \frac{(r_{ij} - \rho(\hat{\theta})_{ij})^2}{q(q+1)}}$$

$\mathrm{RMR} = 0$ indicates perfect fit and because it based on the sample covariance matrix, RMR values and range depend on scale of observed variables. Thus, the results of RMR may change when the scales of the observed variables changed. Therefore, SRMR, which designed to computer on correlation rather than covariances, which can be seen as a standardized version of the RMR, will be more stable when the scales of the observed variables vary.

1.2.2 Incremental Fit Indices

Incremental fit indices, also known as comparative or relative fit indices, are a group of indices that do not use the chi-square in its raw form but compare the chi-square value to a baseline model. With some exceptions, (i.e. latent growth model) the null hypothesis is that all variables are uncorrelated (there are no latent variables). The first of these indices to appear in LISREL output is the Normed Fit Index (NFI: Bentler and Bonnet, 1980). The Comparative Fit Index (CFI: Bentler, 1990) is a rescaled version of the NFI, the values of NFI can fall outside the 0-1.0 range. The NFI tends to underestimate fit in small samples and therefore, Bentler (1990) reversed NFI to take sample size into account and suggest the CFI. The NNFI/TLI (Bentler and Bonnet, 1980) imposes a greater relative penalty for model complexity than the CFI, but only one of these two fit statistics should be reported because their values are highly correlated (Kenny, 2014). The Bollen's Incremental Fit Index (IFI, also called BL89, Bollen, 1989), is relatively insensitive to sample size. Their recommended threshold levels are listed in table 1.2.2.

Table 1.2.2 Incremental Fit Indices and their acceptable thresholds (Moss, 2009)

Fit Index	**Acceptable Threshold Levels**	**Description**
NFI	Values greater than 0.95	Assesses fit relative to a baseline model which assumes no covariances between the observed variables. Often underestimated when samples are small. The fit can be overestimated if the number of parameters is increased. Can be resolved in NNFI (TLI).
NNFI (TLI)	Values greater than 0.95	Non-normed, values can fall outside the 0-1 range. Favours parsimony. Performs well in simulation studies
CFI	Values greater than 0.95	Normed, 0-1 range.
IFI	Values greater than 0.90	Non-normed, values can fall outside the 0-1 range.

Most of these fit indices are calculated by using ratios of the model chi-square and the null model chi-square taking into account their degrees of freedom. All of these indices have values that range between approximately 0 and 1.0 where 1.0 is the best result. Some indices are “normed” so that their values cannot be below 0 or above 1 (e.g., NFI, CFI). Others are considered “nonnormed” because they may be larger than 1 or slightly below 0 (e.g., NNFI, IFI).

The Bentler CFI is another incremental fit index that is also a goodness-of fit statistic. The CFI compares the amount of departure from close fit for the researcher’s model against that of the independence (null) model.

$$\widehat{\Delta}_M = \max(0, \chi_M^2 - df_M) \text{ for compared model}$$

$$\widehat{\Delta}_B = \max(0, \chi_B^2 - df_B) \text{ for baseline model}$$

$$\mathrm{CFI} = 1 - \frac{\widehat{\Delta}_M}{\widehat{\Delta}_B}$$

The value of χ_B^2 is often relatively large and $\chi_B^2 \leq df_B$ is not usually seen in real data. The result $\mathrm{CFI} = 0.90$, says that the fit of the compared model is about 90% better than that of the baseline model.

1.2.3 Parsimony Fit Indices

Parsimony fit indices are designed to penalize models that are less parsimonious, which means simpler models are favored over complex models. When the model becomes more complex, it will fit better to the sample data, however, researchers risk overfitting when complexity is not considered in the model-fitting process (Preacher 2006). To overcome this problem, four parsimony of fit indices have been developed (Hooper, 2008): (1) the Parsimony Goodness-of-Fit Index (PGFI, based on GFI), (2) the Parsimonious Normed Fit Index (PNFI, based on NFI), (3) the Parsimony Incremental Fit Index 2 (PNFI2, based on Bollen's IFI), (4) the Parsimony Comparative Fit Index (PCFI, based on CFI).

Among these indices, the PGFI is based upon the GFI by adjusting for loss of degrees of freedom. The function of PGFI is:

$$\mathrm{PGFI} = \frac{df_M}{df_B} \times \mathrm{GFI} = \frac{2 \times df_M}{k(k+1)} \times \mathrm{GFI}$$

where df_M is the degrees of freedom for the tested model and $df_B$is the degrees of freedom for the null model, k is the number of observed variables.

The PNFI also adjusts for degrees of freedom and it is based on the NFI:

$$\mathrm{PNFI} = \frac{df_M}{df_B} \times \mathrm{NFI} = \frac{2 \times df_M}{k(k+1)} \times \mathrm{NFI}$$

Because the penalty for model complexity, after adjust for degrees of freedom, the values of parsimony fit index are lower or equal to the values unadjusted indices ($\frac{df_M}{df_B} \leq 1$). For example, comparing the PGFI and the GFI, in a particular case, the PGFI can give different results while the GFI fits well because the ratio of $\frac{2 \times df_M}{k(k+1)}$ is large. So far, no confirmed threshold levels have been recommended for these indices, (one recommendation from Mulaik et al 1989 is to obtain parsimony fit indices within the 0.50 region while other goodness of fit indices achieve values over 0.90). This variation makes these Parsimony fit indices difficult to interpret.

"Information criteria" indices like the Akaike Information Criterion (AIC), the Consistent Version of AIC (CAIC, which adjusts for sample size, Akaike, 1974), and Bayesian information criterion (BIC: Schwarz, 1978) can be seen as other forms of parsimony fit indexes.

$$\text{AIC} = \chi_M^2 + 2q$$

$$\text{BIC} = \chi_M^2 + q \times ln(N)$$

where q is the number of free model parameters and N is the sample size.

Like the parsimony fit indices, results of these indices are also adjusted for number of parameters or sample size. For example, AIC adjusts the result of chi-square test using number of free parameters in the model, this change is a function of model complexity. And the relative correction for complexity of the AIC becomes smaller as the sample size increases, thus BIC includes sample size in the adjustment.

These indices are generally used when comparing models estimated with the same data. Smaller values suggest a better fit. Generally, "information criteria" indices will not be used when there is only one model, because it's difficult to suggest a cut-off. Additionally, these indices need a sample size of 200 or more to perform well (Homburg, 1991).

1.2.4 Reporting Fit Indices

While there is no consensus on rules for assessment of model fit, using a variety of indices in research is necessary because different indices reflect a different aspect of model fit. However, it will be a burden both for the researcher and reader if the researcher includes every index in their reports. Choosing the most appropriate fit indices and not the indices that indicate the best fit for a particular situation is important. When the most appropriate fit indices are undefined, one suggestion is using combinations of fit indices. For example, Hu and Bentler, 1999 suggested a two-index presentation format, and the commonly used combinations are listed in Table 1.2.4.

Table 1.2.4 Two-Index Presentation Strategy

Fit Index Combination	**Combinational Rules**
NNFI (TLI) and SRMR	NNFI $\geq$ 0.95, SRMR $\leq$ 0.08
RMSEA and SRMR	RMSEA $\leq$ 0.05, SRMR $\leq$ 0.08
CFI and SRMR	CFI $\geq$ 0.95, SRMR $\leq$ 0.08

1.3 The Inadequacy of Traditional Fit Indices

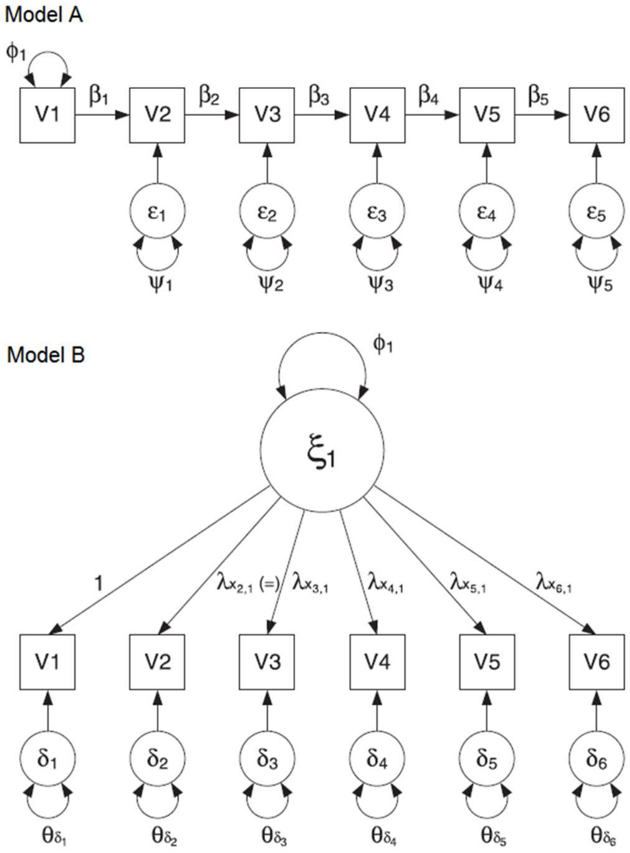
Degrees of freedom and sample size are often used to adjust fit indices. In most cases, potential SEM models are built on the same sample data covariances or correlation matrix. Thus, sample size for all these models are either the same or approximately the same. The most important factor that penalizes more complex models is the degrees of freedom. Many traditional fit indices are already adjusted by degrees of freedom or number of free model parameters. For example, SRMR, PGFI are adjusted by model degrees of freedom and AIC, BIC are adjusted by number of free model parameters.

In SEM, the number of free model parameters (q) instead of degrees of freedom of a SEM model (df) is more often used to indicate model complexity. It is defined as the number of free parameters minus the number of functional constraints placed in a model. Generally, with all else being equal, models with larger q (small df) are considered as more complex (Kenny, 2014).

However, there is no guarantee that model complexity is accounted for by the number of free parameters. In fact, evidence indicates the inadequacy of traditional fit indices (Preacher, 2006). Preacher, 2006 showed that some SEM models performed better than the others using the same fit index, even though these models had same number of free parameters. I define fitting propensity (FP) as a model's average ability to fit different data patterns, when all else is equal. That is, two SEM models may have different FP, even though they have the same number of free model parameters.

Preacher (2006), built two models that have the same number of free model parameters (Model A and B showed in Figure 1.3.1, both have q =11).

Figure 1.3.1 Model A and Model B from Preacher, 2006 (*q* =11)

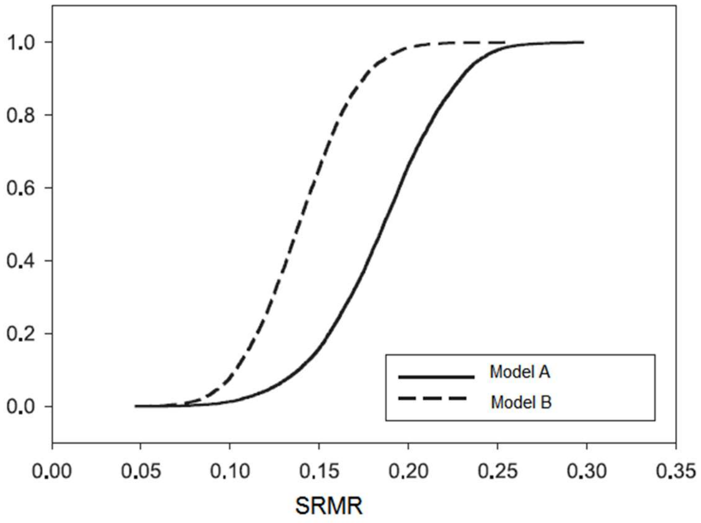


To test their complexity, 10,000 random data sets (10,000 6 × 6 correlation matrices) were generated. Then Model A and Model B were used to fit these data sets, fit index SRMR was used to detect how these models fit the generated data. As mentioned in section 1.2.1, SRMR is the standardized version of the RMR, it is calculated as:

$$SRMR = \sqrt{\frac{tr(S - \Sigma(\hat{\theta}))^2}{p(p+1)}}$$

A good fit occurs if the value of SRMR is less than 0.08. Cumulative frequency distribution (CDF) of SRMR for both Model A and Model B were showed in Figure 1.3.2.

Figure 1.3.2 CDFs of SRMR for Model A and Model B from Preacher, 2006



Results showed that Model B fit random samples better than Model A, even though they had same number of free parameters. This is strong evidence to show the complexity of models is not fully controlled by the number of free parameters. Thus, more factors that can reflect some characters of the SEM model should be identified and used to either adjust the traditional fit indices or to create new fit indices. For example, the confirmatory tetrad analysis (Bollen, 1993) has been proposed as one way to build SEM models and potentiallyprovide a reasonable way to improve the measurement of model complexity.

1.4 Confirmatory Tetrad Analysis

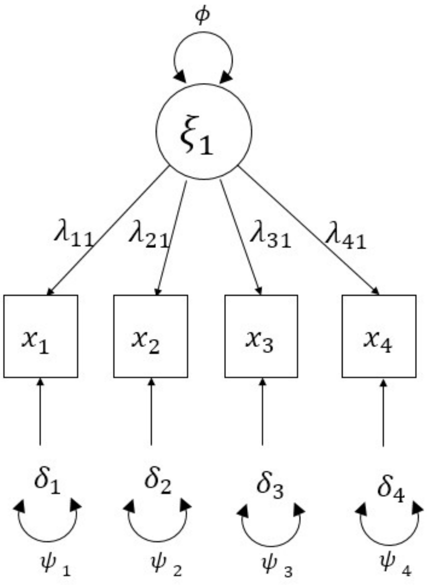
Confirmatory tetrad analysis (CTA) is a technique to estimate model fit of structural equation models by using the features of tetrads, first proposed by Spearman (1904). The tetrad approach was used in SEM model testing but has been replaced by the maximum likelihood method popularized by Jöreskog (1970) in the LISREL program. Due to the development of computers, Glymour et al. (1987) proposed vanishing tetrads as a method to search for SEM models, the proposed exploratory tetrad analysis (ETA) was based on a computer intensive search algorithm. And then, confirmatory tetrad analysis (CTA) was developed by Bollen & Ting (1993) to test one of several specific SEM models. After a set of Stata commands for CTA was developed by Bauldry & Bollen (2016), the CTA has become more accessible.

Like all the other techniques in SEM, the primary goal of CTA is to test $H_0\colon \Sigma = \Sigma(\theta)$. However, CTA focuses on the vanishing tetrads implied by the proposed model rather than the estimation of coefficients in the function $\mathbf{y} = \mathbf{B}\mathbf{y} + \boldsymbol{\Gamma}\mathbf{x} + \boldsymbol{\zeta}$. In Bollen & Ting (1993), a "tetrad" refers to the difference in the products of certain covariances (or correlations) among four random variables. A structural equation model often implies that some tetrads should be zero, and this model indicated that zero tetrads are called "vanishing tetrads". Thus, by testing the model implied vanishing tetrads equal to zero or not, the goodness of fit of the model can be determined.

Compared to the traditional model fit indices in SEM, the potential benefits from CTA are the following:, 1) CTA can be applied to some under-identified models (at least one parameter in function $\mathbf{y} = \mathbf{B}\mathbf{y} + \boldsymbol{\Gamma}\mathbf{x} + \boldsymbol{\zeta}$ that cannot be consistently estimated) and some models that are not nested according to the traditional LR test are nested in terms of vanishing tetrads. 2) there is no parameter estimate process in CTA, which means CTA does not require numerical minimization and thus avoids the associated convergence problems that are present with other estimation approaches.

The example below is designed to illustrate the concept of tetrad and vanishing tetrad.

Figure 1.4.1 Path diagram of a factor model



The factor model shown in Figure 1.4.1 has one latent variable ξ_1 and four observed variables x_1 to x_4. The equations corresponding to this factor model are of the form:

$$x_i = \lambda_{i1}\xi_1 + \delta_i \tag{1.4.1}$$

where δ_i is the random error term with $E(\delta_i) = 0$ for all i, COV(δ_i, δ_j) = 0 for $i \neq j$, and the COV(ξ_1, δ_i) = 0 for all i. Then the population covariances (σ_{ij}) of the observed variables can be calculated by the following form:

$$\sigma_{ij} = \lambda_{i1}\lambda_{j1}\phi \tag{1.4.2}$$

where σ_{ij} is the population covariance of the i and j variables and ϕ is the variance of ξ_1. If the model is correct, then we can use covariance algebra (e.g., Bollen 1989, p. 21) to prove that the equalities below must hold:

$$
\begin{aligned}
\tau_{1234} &= \sigma_{12}\sigma_{34} - \sigma_{13}\sigma_{24} = \lambda_{11}\lambda_{21}\phi * \lambda_{31}\lambda_{41}\phi - \lambda_{11}\lambda_{31}\phi * \lambda_{21}\lambda_{41}\phi = 0 \\
\tau_{1342} &= \sigma_{13}\sigma_{42} - \sigma_{14}\sigma_{32} = 0 \\
\tau_{1423} &= \sigma_{14}\sigma_{23} - \sigma_{12}\sigma_{43} = 0 \qquad (1.4.3)
\end{aligned}
$$

A τ_{ghij} shown in 1.4.3 is called a "tetrad" in confirmatory tetrad analysis, and when τ_{ghij} is zero for a model, it is so called a "vanishing tetrad". From the results in 1.4.3, the factor model in Figure l.4.1 implies three vanishing tetrads (τ_{1234}, τ_{1342} and τ_{1423}). If the construction of SEM model changed, the composition of vanishing tetrads will also change. Similar to the model showed in Figure 1.4.1, the factor model showed in Figure 1.4.2 also has 4 observed variables (x_1 to x_4), however, it has 2 latent variables (ξ_1 and ξ_2).

Figure 1.4.2 Path diagram of a factor model with two latent variables

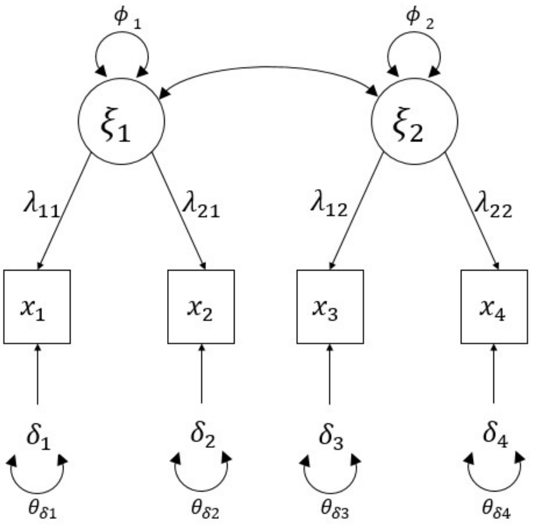


Figure 1.4.2 showed a different path diagram for the same observed variables (x_1 to x_4) in Figure 1.4.1. We use $\sigma = COV(\xi_1, \xi_2)$ to represent the covariance between ξ_1 and ξ_2, it is not equal to zero. Thus, the 6 population covariances (σ_{ij}) of these 4 observed variables are calculated as:

$$\sigma_{12} = \lambda_{11}\lambda_{21}\phi_1 \qquad \sigma_{34} = \lambda_{12}\lambda_{22}\phi_2$$

$$\sigma_{13} = \lambda_{11}\lambda_{12}\sigma \qquad \sigma_{14} = \lambda_{11}\lambda_{22}\sigma$$

$$\sigma_{23} = \lambda_{21}\lambda_{12}\sigma \qquad \sigma_{24} = \lambda_{21}\lambda_{22}\sigma$$

In this factor model, the assumption assumed that $\phi_1 \neq \phi_2$, thus, there is only one vanishing tetrad implied in this model:

$$\tau_{1342} = \sigma_{13}\sigma_{42} - \sigma_{14}\sigma_{32} = \lambda_{11}\lambda_{12}\sigma * \lambda_{21}\lambda_{22}\sigma - \lambda_{11}\lambda_{22}\sigma * \lambda_{21}\lambda_{12}\sigma = 0 \quad (1.4.4)$$

Algebraic substitution between vanishing tetrads will show that some of the vanishing tetrads can be derived from the others and are redundant for the test. Therefore, not all the vanishing tetrads should be used in the test of overall model fit in CTA, and the identification of nonredundant vanishing tetrads is necessary before the test of model it. There are three situations in which redundancy will occur: (1) When none of the covariances exist in one varnishing tetrad are exist in other vanishing tetrads: Algebraic substitution is impossible, and this tetrad is nonredundent. (2) When two vanishing tetrads have three or more covariances in common: They must be identical, thus they are redundant. (3) Vanishing tetrads having one or two covariances in common: We need to distinguish the redundant/nonredundant tetrads (Bollen, 1993).

For vanishing tetrads having two covariances in common, we always have 3 different tretrads for one choice of 4 observed variables. For example, in Example shown in Figure 1.4.1, we have:

$$\tau_{1234} = \sigma_{12}\sigma_{34} - \sigma_{13}\sigma_{24} = 0$$

$$\tau_{1342} = \sigma_{13}\sigma_{42} - \sigma_{14}\sigma_{32} = 0$$

$$\tau_{1423} = \sigma_{14}\sigma_{23} - \sigma_{12}\sigma_{43} = 0$$

We can see any 2 of these 3 equations can imply the third equation.

When τ_{1234} and τ_{1342} are true, from τ_{1234} , we know $\sigma_{12}\sigma_{34} = \sigma_{13}\sigma_{24}$, then replace $\sigma_{13}\sigma_{42}$ using $\sigma_{12}\sigma_{34}$ in τ_{1342}, we have $\sigma_{12}\sigma_{34} - \sigma_{14}\sigma_{32} = 0$, which is the same as τ_{1423}. Thus τ_{1423} is a redundant vanishing tetrad that need to be removed before significance test.

For vanishing tetrads having one covariance in common, algebraic substitution will lead to a vanishing equation with six covariances, and no additional vanishing tetrad will be implied.

For example, in a SEM model, we have:

$$\tau_{1235} = \sigma_{12}\sigma_{35} - \sigma_{13}\sigma_{25} = 0$$

$$\tau_{1264} = \sigma_{12}\sigma_{64} - \sigma_{16}\sigma_{24} = 0$$

$$\tau_{1635} = \sigma_{16}\sigma_{35} - \sigma_{13}\sigma_{56} = 0$$

From τ_{1235} and τ_{1264} that have one common covariance in common (σ_{12}) we know:

$$\sigma_{12} = \sigma_{13}\sigma_{25}/\sigma_{35}$$

Then put this in τ_{1264}, we will get:

$$\sigma_{13}\sigma_{25}\sigma_{64} - \sigma_{16}\sigma_{24}\sigma_{35} = 0$$

Thus this equation and τ_{1635} implied:

$$\sigma_{16}\sigma_{35} = \sigma_{13}\sigma_{56}$$

$$\sigma_{13}\sigma_{25}\sigma_{64} - \boldsymbol{\sigma_{16}}\sigma_{24}\boldsymbol{\sigma_{35}} = \sigma_{13}\sigma_{25}\sigma_{64} - \boldsymbol{\sigma_{13}\sigma_{56}}\sigma_{24} = \sigma_{25}\sigma_{64} - \sigma_{56}\sigma_{24} = \tau_{2456}$$

That means given vanishing tetrads τ_{1235}, τ_{1264} and τ_{1635}, the vanishing tetrad τ_{2456} is redundant. Thus, before we compute the test statistic, a set of nonredundant vanishing tetrads must be selected. For example, model in Figure 1.4.1 we can select 3 different nonredundant vanishing tetrad sets. It is possible that difference sets might yield different results, thus Hipp and Bollen (2003) recommended randomly selecting sets of vanishing tetrads multiple times and assessing the sensitivity of the results to different selections.

After model implied nonredundant vanishing tetrads are determined, a simultaneous significance test showed below was proposed by Bollen (1990) and it can be used to determine whether the model is consistent with the sample data (covariances or correlation matrix).

$$\sqrt{N}\boldsymbol{t} \overset{D}{\rightarrow} N(0, \Sigma_{tt})$$

$$\Sigma_{tt} = (\partial\boldsymbol{\tau}/\partial\boldsymbol{\sigma})'\Sigma_{ss}(\partial\boldsymbol{\tau}/\partial\boldsymbol{\sigma})$$

$$T = N\boldsymbol{t}'\hat{\Sigma}_{tt}^{-1}\boldsymbol{t} \sim \chi_t^2 \tag{1.4.5}$$

where N is the sample size.

$\boldsymbol{t}$ is the column vector of the independent tetrad differences.

Σ_{tt} is the covariance matrix of the limiting distribution of the sample tetrad differences.

Σ_{ss} is the covariance matrix of the limiting distribution of the sample covariances appear in the sample tetrad differences.

$\boldsymbol{\tau}$ is a vector of the population tetrads that are implied to be zero for a specific model.

$\boldsymbol{\sigma}$ is a column vector of all unique covariances appear in the population tetrads.

t is the number of population tetrads.

Use the vector $\boldsymbol{\tau}$ to represent the population tetrads that are implied to be zero for a specific model in 1.4.5, model goodness of fit can be tested by evaluating whether all model implied vanishing tetrads are equal to zero based on the sample covariance matrix instead of evaluating how well the model implied covariance matrix can match the sample covariance matrix. The null hypothesis of test in 1.4.5 is $\mathrm{H}_0\colon \boldsymbol{\tau} = 0$, instead of $\mathrm{H}_0\colon \Sigma = \Sigma(\theta)$. Test statistic T will approximates a chi-square variate with degrees of freedom equal to the number of tetrad differences simultaneously examined. A nonsignificant test statistic means that the implied vanishing tetrads hold and the model is reasonable. If a significance test result was found, the model should be rejected.

This simultaneous test statistic for multiple vanishing tetrads used in CTA can be applied to normally or nonnormally distributed observed variables. Testing vanishing tetrads provides a test for model fit that can lead to results different from the usual likelihood-ratio (LR) test associated with the maximum likelihood methods that dominate the structural equation field (Bollen & Ting, 1993). Also, as mentioned before, some models that are not nested according to the traditional LR test (nested in parameters) are nested in terms of vanishing tetrads, CTA may be a good choice to do the comparison in these cases. For example, the vanishing tetrad implied by the model in Figure l.4.2 is a subset of the vanishing tetrads implied by the model in Figure l.4.1, thus, these two models have "nested tetrads." If the difference in the test statistics for the two models is not significant, this lends support to the model that implies the more vanishing tetrads. If the test result is significant, the model having fewer vanishing tetrads is preferred (Bollen, 1993). Therefore, just like number of free parameters, number of vanishing tetrads can be seen as another factor to determine the complexity of SEM models.

Based on these characters, number of vanishing tetrads appears to have potential to complement traditional likelihood methods to test the model goodness of fit, and number of vanishing tetrads may be another index of model complexity.

1.5 Summary

In section 1.3, we indicated an inadequacy of SEM fit indices, that is, in commonly used fit indices, only model degrees of freedom/number of free parameters is used to penalize the model complexity. However, the complexity of a model is not determined solely by the degrees of freedom. Preacher (2006) examined the relationship between model complexity and the number of free parameters and, after comparing SEM models that have same number of free parameters, he confirmed that models may have different model complexity even though they have same number of free parameters. These differences exist because they have different constraints or different functional forms. It becomes clear these kinds of differences cannot be distinguished by model degrees of freedom/number of free parameters, thus, these differences were ignored in the traditional fit indices.

Confirmatory tetrad analysis (CTA) is an alternative and potentially complementary method of testing and comparing the fit of SEM models to the commonly used likelihood ratio tests (Bollen 2016), models not nested in traditional approaches may nested in terms of their vanishing tetrads. Furthermore, the vanishing tetrad numbers of a model is not only based on the number of observed variables, but also affected by the constraint types and functional form. Thus, vanishing tetrad numbers may be a good method for penalizing complex model constrains and complex functional forms.

On the other hand, when comparing SEM models that have "nested" vanishing tetrads in CTA, models that has larger vanishing tetrad numbers are considered the restrictive model ($t_R > t_F$), and if there is no significant difference in the CTA test results, the model with largest vanishing tetrad numbers will be retained. This is very similar to the comparison of nested linear models. In linear regression, the restricted model is obtained from the full model, so restricted model has larger model degrees of freedom than full model ($df_R > df_F$). The restrictive model is preferred if there is no significant difference between them. Thus, the role of vanishing tetrad number in CTA is very similar to the role of model degrees of freedom in linear regression. This provides strong support of our hypothesis that vanishing tetrad numbers can be another indicator of SEM model complex.

The main purpose of this dissertation is to examine whether the SEM model complexity is related to the number of vanishing tetrads implied in the models. To determine model complexity, large numbers of plausible random data sets (correlation matrices) are needed to represent the full data space. In this study, two kinds of random data were generated using R version 3.3.1. One is uniform random correlation matrices and the other is known-model random correlation matrices. For uniform random correlation matrices, several data generation methods were developed and compared, with the most suitable one being used to examine tetrad numbers ability to account for model complexity. To demonstrate that vanishing tetrad number is an indicator of model complexity, several model pairs that have the same number of free parameters but different vanishing tetrad numbers will be fit to the generated data. It is hypothesized that when models have the same model degrees of freedom, model that has largest vanishing tetrad number will be the most complex model.

The second purpose of this dissertation is to find the situations in which the vanishing tetrad number serves as a complement to model degrees of freedom to indicate SEM model complexity. Because in many situations, model complexity cannot all indicated by number of vanishing tetrads. For example, when some constraints are added to a SEM model and changed model complexity, its number of vanishing tetrad will not change while its model degrees of freedom do change. Thus, vanishing tetrad numbers should be used in specific situations to improve the evaluation of model complexity. It is hypothesized that when models are nested in terms of tetrads, vanishing tetrad number can be helpful to indicate which model is more complex.

The third purpose of this dissertation is to create an R program to conduct the Confirmatory Tetrad Analysis, including the calculation of model implied vanishing tetrad numbers using the empirical method, the CTA test (chi-square test) result and several fit indices (e.g. RMSEA, RMR, SRMR, AIC, BIC). Other R programs will be created to generate the random data set and to do the comparison of model complexity.

Chapter 2 Methods

2.1 Two Methods to Assess SEM Model Complexity

To examine the tetrad number as a new indicator of model complexity, one need to access the relationship between tetrad numbers and model complexity. As mentioned in Section 1.3, model complexity is defined as a model's average ability to fit different data patterns. Thus, in order to investigate the complexity of SEM models, large numbers of computer-generated random correlation matrices were needed in this study. The idea of using computer-generated random correlation matrices to investigate the complexity of SEM models was proposed by Collyer (1985) and already successfully used in Preacher's evaluation of SEM model complexity (Preacher, 2003).

This study included two different methods to assess SEM model complexity, and they use two different kinds of random data. One uses the uniform random correlation matrices, the other one uses the known-model random correlation matrices, both defined by Collyer (1985).

2.1.1 Uniform Random Correlation Matrices

In order to investigate SEM model complexity, large numbers of uniform random correlation matrices will be needed (Preacher, 2006). Uniform random correlation matrices defined by Cutting (2000) as matrices where every possible square, symmetric, positive semidefinite matrix with 1s on the diagonal and off-diagonal elements in the range {0, 1} has an equal probability of being selected. Negative values are rarely seen in sample covariance matrices, and in this dissertation, we prefer to emphasis on fitting models to "plausible" data rather than to “possible” data (Roberts and Pashler, 2000). Furthermore, the requirement of positive semi-definite makes the elements in correlation matrices are not uniformly distributed. However, correlation matrices generated according to these criteria can be seen as uniformly distributed because (1) they do not presume a particular model and (2) every possible matrix fitting the criteria has an equal chance of being selected (Preacher 2003). It is necessary to use uniformly distributed random correlation matrices in this study because the density plot of fitting results, the cumulative frequency distribution (CDF) plot of fitting results and the frequency of data sets fit well/better by a SEM model when apply to these random correlation matrices are a representation model complexity. If the random correlation matrices are not uniformly distributed, bias would occur. Some existing methods of generating uniform random correlation matrices are discussed in section 2.2. Each method will be described in terms of computational efficiency and representativeness.

2.1.2 Known-model Random Correlation Matrices

The other type of random correlation matrices used to examine the complexity of competing SEM models is Known-model random population correlation matrices. Known-model random population correlation matrices were generated using each of the competing SEM models by randomly generate the values of linear coefficients in the model (random values between 0.05 and 0.95), then fixing the variances for each variable equal to 1 by adjusting the variance of random error terms. When developing SEM models, linear coefficients with value close to 0 or 1 are rarely seen, thus on the purpose to emphasis on fitting models to "plausible" data rather than to "possible" data, range (0.05, 0.95) was used instead of range (0.00, 1.00).

Using different sets of parameter values, different Known-model population correlation matrices can be generated based on Equation 1.1.4 and 1.1.8, since all elements in the model implied population correlation matrix are related to the parameters and random error terms. Thus, in Know-model analysis, if a model can fit random population correlation matrices that are generated by its competitor model, it should be seen as a more complex model than its competitor model (Collyer, 1985).

2.2 Methods to Generate Uniform Random Correlation Matrices

2.2.1 The Uniform Correlation Matrix (UCM) Method

The uniform correlation matrix method, also called the direct acceptance-rejection method is used for generating random correlation matrices that proposed by Botha et al. (1988). To generate a $p \times p$ random correlation matrix, the first step is generating $\frac{1}{2}p(p-1)$ random numbers which follow a uniform (0, 1) distribution. The second step is putting them to the upper triangle of a $p \times p$ matrix and transposing the upper triangle elements to the lower triangle. After setting all the diagonal elements equal to 1, a random matrix is generated. This random matrix should be retained in this study if it is positive definite, otherwise it should be excluded. Thus the third step is check the eigenvalues of a generated random matrix. If there are p positive eigenvalues for a $p \times p$ matrix (all eigenvalues are positive), this matrix can be retained as a random correlation matrix.

Botha et al. found that the UCM method is simple and effective when the dimension of a matrix is low ($p \leq 6$). However, when the dimension increases, it becomes harder for the UCM method to find a positive definite matrix from the generated matrices. This is shown by our examination of the UCM method. In our findings, when $p = 5$, the UCM method takes 2 seconds for a Core i5 6500 processor to generate 1000 positive definite matrices using R 3.3.1. When $p = 6$, UCM method takes 11 seconds to generate 1000 retained matrices because approximately one out of every 50 matrices is positive definite. And when p is larger than 7, the UCM method slows down rapidly. When $p = 8$, it takes 3 seconds to generate 1 positive definite matrix and when $p = 10$, it takes 100 seconds to generate 1 positive definite matrix. Because this study needs a large amount of random correlation matrices (n>10000) with dimensions at times greater than 10, the UCM method should not be used in these comparisons.

Therefore, generating methods are faster and more reasonable than UCM, and will be the focus of this dissertation.

2.2.2 The Markov Chain Monte Carlo Method

The use of Markov Chain Monte Carlo (MCMC) method has become very common in evaluating statistical estimators for structural equation models. To generate a $p \times p$ positive definite correlation matrix, MCMC method has the same logic as the UCM method. Like UCM method, the first step is to generate $\frac{1}{2}p(p-1)$ random numbers on the interval {0, 1}, put them in the upper triangle of the matrix, then transpose them to the lower triangle, and after setting all the diagonal elements equal to 1 a random correlation matrix is generated. The difference between MCMC method and UCM method is their ways to find positive definite correlation matrix among these generated random correlations.

To generate positive definite random correlation matrices using MCMC method, the first step is set a square, symmetric, positive definite (or nearly positive definite) matrix as the starting matrix. At the first iteration, the algorithm uses the starting matrix and perturbs the off-diagonal elements, retaining symmetry to generate a new random correlation matrix. Then the new generated matrix is checked to see if it is positive definite. If it is not positive definite, the matrix is ignored and another random matrix is generated; the staring matrix is used again as the starting point. When a new positive definite random correlation matrix is found, this new matrix will be retained and chosen as the starting point for the next iteration. After a sufficient number of iterations (the burn-in phase), the distribution of the retained matrices approximates the target distribution of random uniform matrices, because there is a dependency between a generated matrix and the matrices that were used to generate it. To minimize the dependency problem, generated matrices are selected and thinned. For example, if the thinning value is set as 10, that means the next retained random correlation matrix will be the matrix after 10 times iteration.

Preacher (2003) compared the Metropolis-Hastings MCMC algorithm with the UCM method. His conclusion is that in the UCM method, the ratio of the number of unacceptable correlation matrices increases exponentially with orders higher than 7 thus is inefficient for this study. The MCMC method will also decrease efficiency when the order increased, but to a far lesser degree. After comparing the generated positive definite random correlations from UCM method and MCMC method, Preacher (2003) concluded that "the MCMC method was judged to provide correlation matrices indistinguishable from those generated by the UCM method, yet at a much faster rate". Thus, the Metropolis-Hastings MCMC algorithm was considered as a potential data generation method in this study.

2.2.3 Partial Correlation Method

Partial correlation method is an effective way to generate positive definite random correlation matrices that functions based on the terms of the correlations and the partial correlations, proposed in Joe (2006). Consider a correlation matrix that represents the correlations between variables x_1, x_2 and x_3. If the correlations between x_1, x_2 and x_1, x_3 are $r_{12} = 0.80$ and $r_{13} = 0.80$ respectively, the first-order partial coefficient of correlations $r_{12.3}$ is

$$r_{12.3} = \frac{r_{12} - r_{13} r_{23}}{\sqrt{(1 - {r_{13}}^2)(1 - {r_{23}}^2)}} \qquad (2.2.2.2)$$

Thus, to ensure $-1 \leq r_{12.3} \leq 1$， r_{23} must between 0.28 and 1.00 (Stanley & Wang 1969).

To generate a positive definite random correlation matrix using partial correlation method, first generate some correlations (e.g. correlations in the first row/column of the matrix) independently in the interval (0, 1). Then use Equation 2.2.2.2 to calculate the range of other allowable correlations and then generate random values in these ranges until all correlations in the matrix are generated.

By doing this, we can generate a positive definite random correlation matrix easily. For example, it takes only 2 seconds for the partial correlation method to generate 1000 positive definite random 10 by 10 correlation matrices meanwhile, the UCM method needs approximate 100 seconds to generate 1 positive definite random 10 by 10 correlation matrix under the same conditions. Therefore, the partial correlation method was also considered as a candidate for the data generation method in this study.

2.3 Determining the Best Method to Generate Uniform Random Correlation Matrices

Random correlation matrices generated by UCM method should be the most suitable random data in this study. However, when p larger than 7, the UCM method will be very slow thus impractical to generate large samples. For random correlation matrices that have p larger than 7, the MCMC method and the Partial Correlation method are potential alternative data generation methods. To compare the methods, their speeds of generation and the distribution of data samples should be considered. In general, the method that takes less time and also can generate random correlation matrices whose distribution match closely to the random correlation matrices generated by UCM method are better for generating random correlation matrices.

Three data generation methods (UCM, MCMC and PCM) were mentioned in Section 2.2, on the purpose of determining the most suitable method for this study, all three methods were used to generate 10,000 uniform random correlation matrices. Random seed 123 and correlation matrices with 4 different dimensions were used (p =4, 5, 6, and 7), in each setting, we measured the data generation speed for each method and compared the correlation matrices to see if MCMC or PCM can yield equally distributed matrices to UCM. To compare the similarity of the matrices, the lower triangle elements were selected from every matrix and used to create histograms and plots of the empirical cumulative distribution function (CDF). Comparison of density plots and CDF plots showed high similarity for UCM and MCMC, but low similarity for Partial Correlation Method.

Figure 2.3.1 Empirical CDF plots of lower triangle elements for UCM(Black), MCMC(Blue), and PCM(Red) algorithms for 4 x 4 matrices.

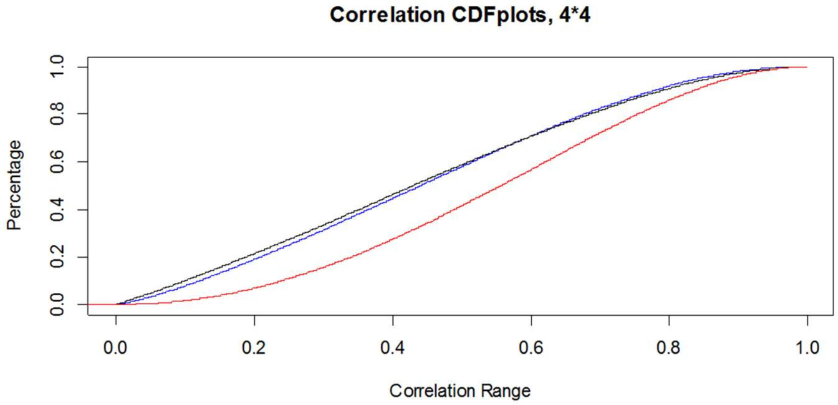


Figure 2.3.2 Empirical CDF plots of lower triangle elements for UCM(Black), MCMC(Blue), and PCM(Red) algorithms for 5 x 5 matrices.

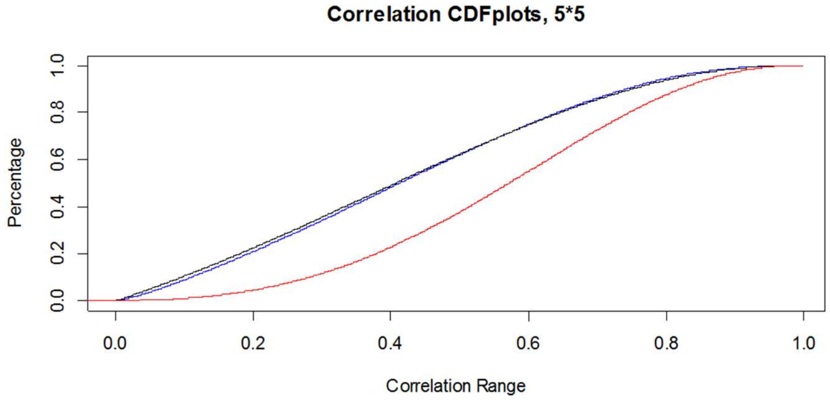


Figure 2.3.3 Empirical CDF plots of lower triangle elements for UCM(Black), MCMC(Blue), and PCM(Red) algorithms for 6 x 6 matrices.

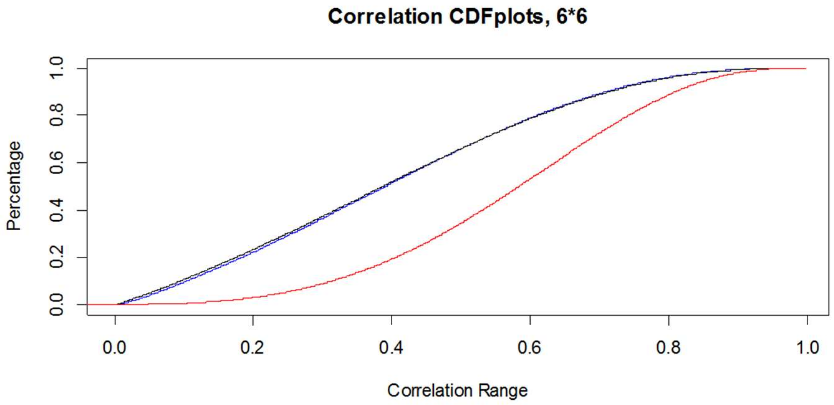


Figure 2.3.4 Empirical CDF plots of lower triangle elements for UCM(Black), MCMC(Blue), and PCM(Red) algorithms for 7 x 7 matrices.

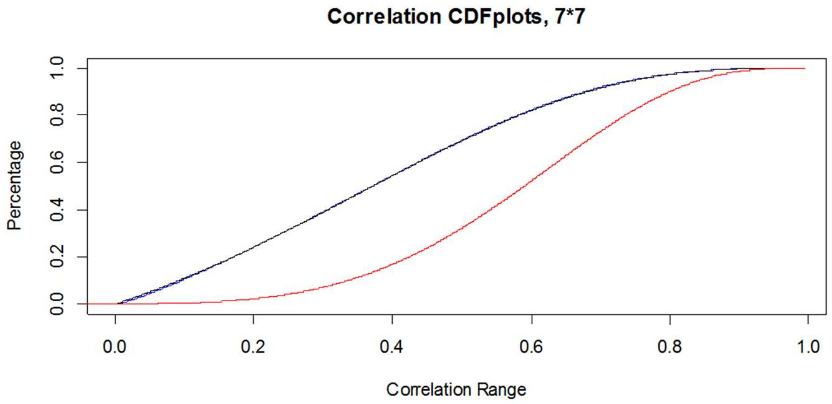


Figure 2.3.5 Histogram of lower triangle elements for UCM(Black), MCMC(Blue), and PCM(Red) algorithms for 4 x 4 matrices.

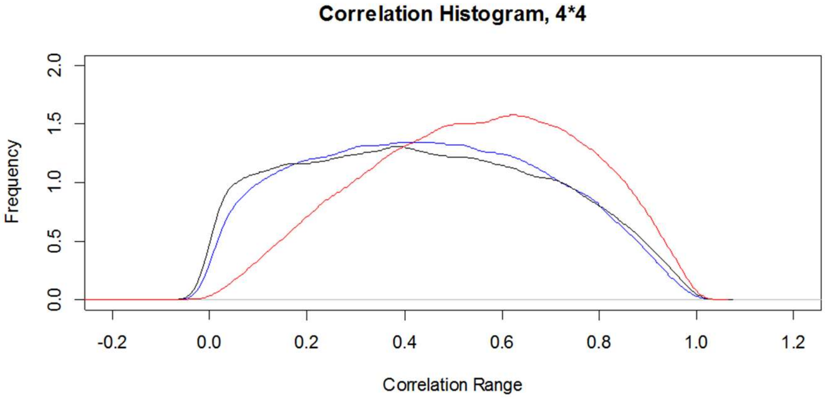


Figure 2.3.6 Histogram of lower triangle elements for UCM(Black), MCMC(Blue), and PCM(Red) algorithms for 5 x 5 matrices.

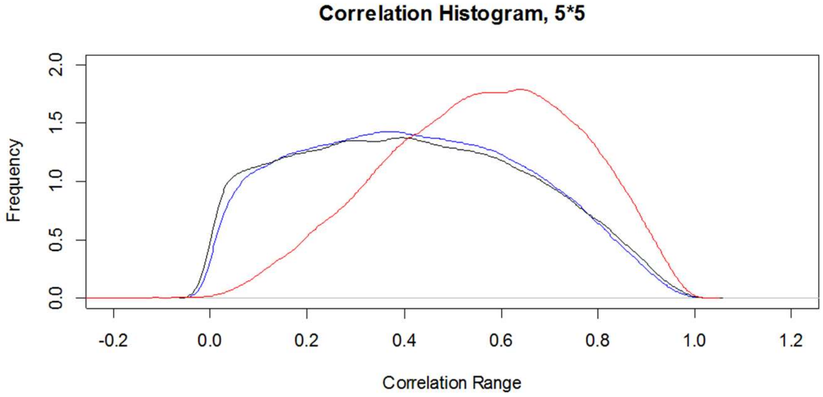


Figure 2.3.7 Histogram of lower triangle elements for UCM(Black), MCMC(Blue), and PCM(Red) algorithms for 6 x 6 matrices.

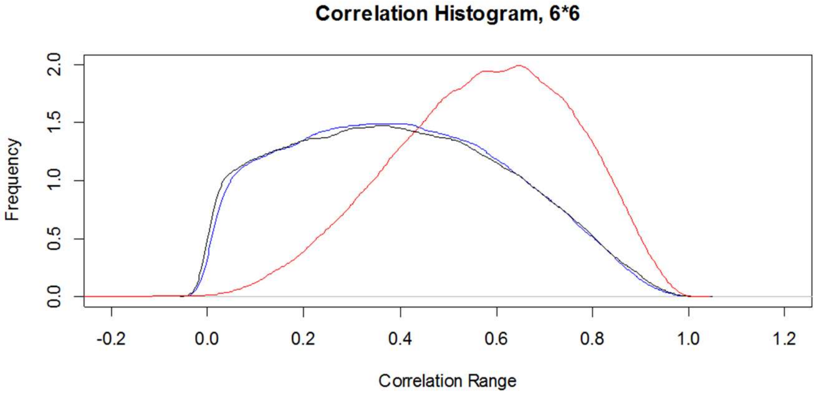
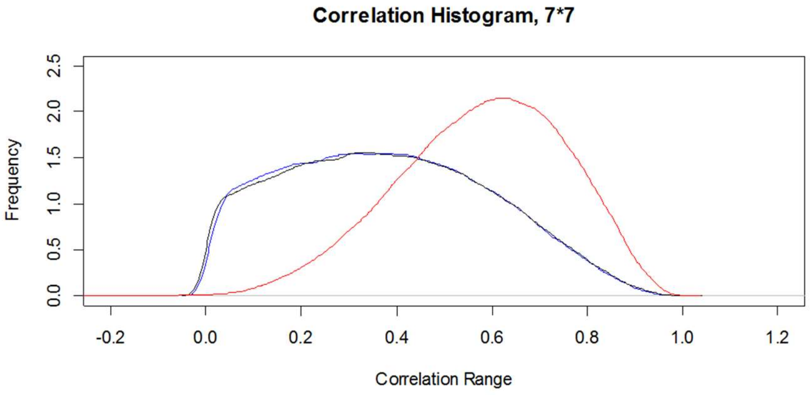


Figure 2.3.8: Histogram of lower triangle elements for UCM(Black), MCMC(Blue), and PCM(Red) algorithms for 7 x 7 matrices.



To further test the similarity of correlation matrices generated from the UCM, MCMC and PCM algorithm, eigenvalues were calculated and from every matrix and ordered from largest to smallest values. Then histograms of these eigenvalues were used to examine the differences between the matrices generated from different data generation methods. From the eigenvalue histograms, we confirmed that UCM and MCMC algorithms generated very similar correlation matrices in every dimension setting while the correlation matrices from PCM algorithm were very different.

Figure 2.3.9: Eigenvalue distribution from UCM, MCMC, and PCM algorithms for 4 x 4 matrices.

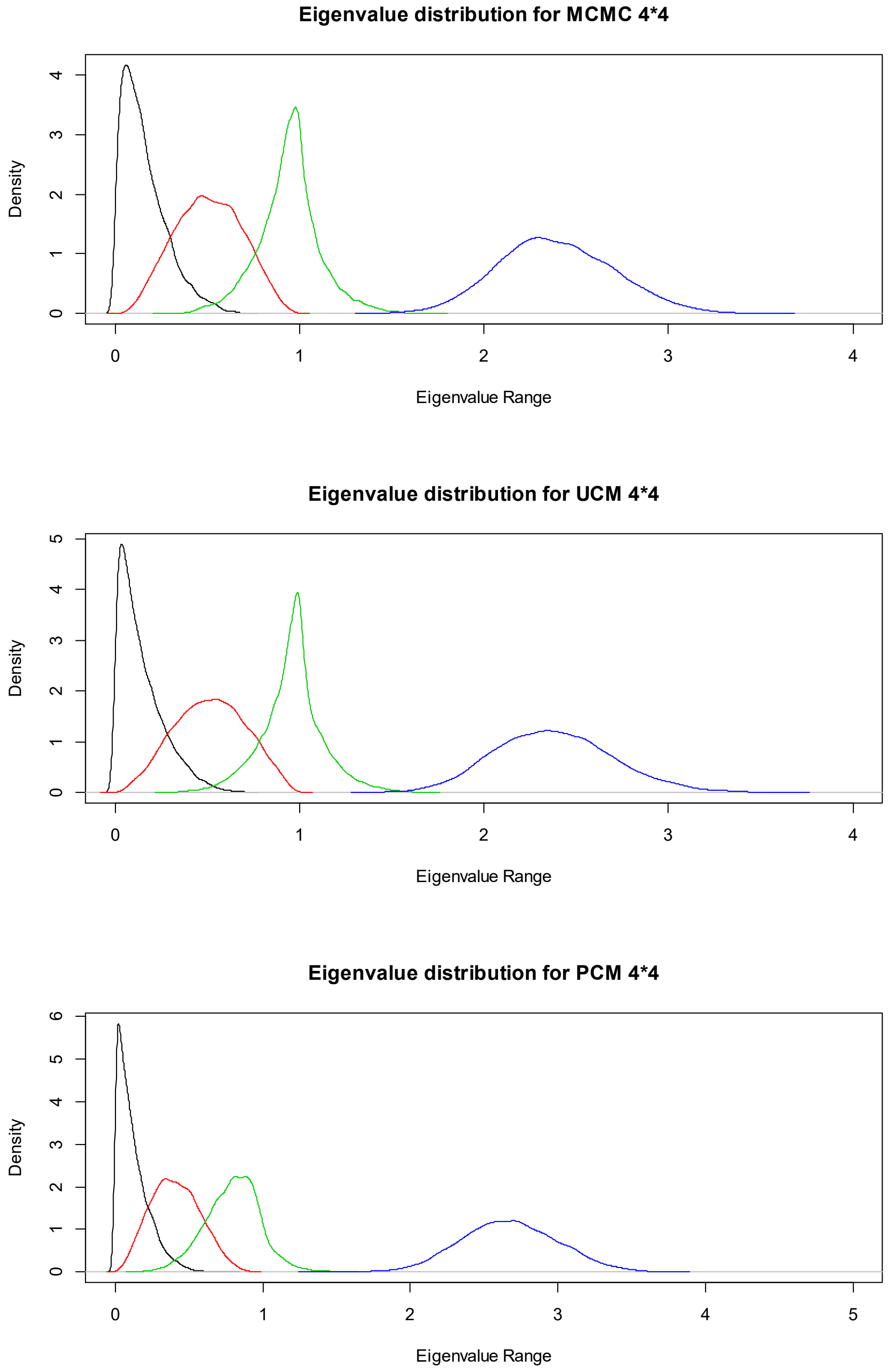


Figure 2.3.10: Eigenvalue distribution from UCM, MCMC, and PCM algorithms for 5 x 5 matrices.

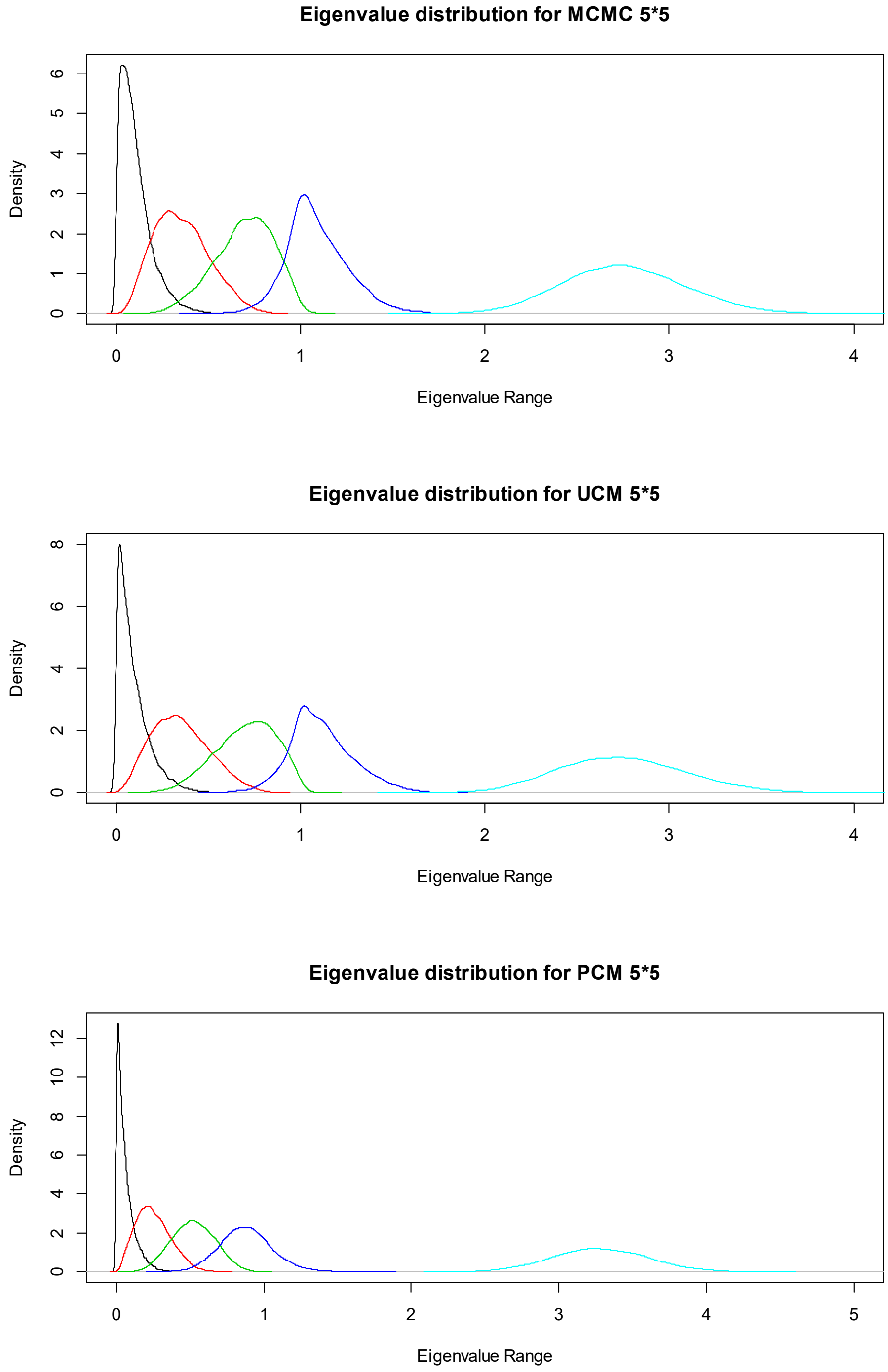
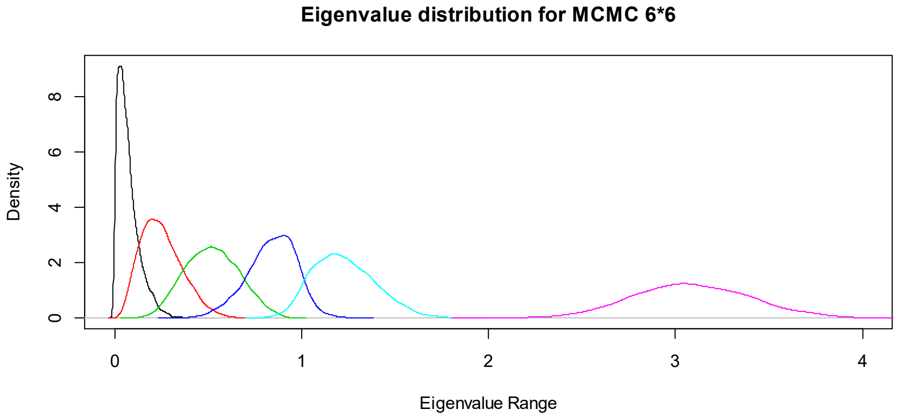
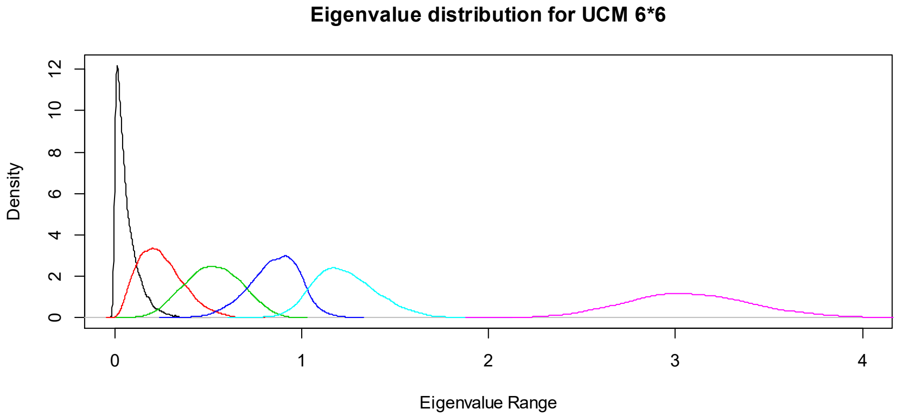


Figure 2.3.11: Eigenvalue distribution from UCM, MCMC, and PCM algorithms for 6 x 6 matrices.





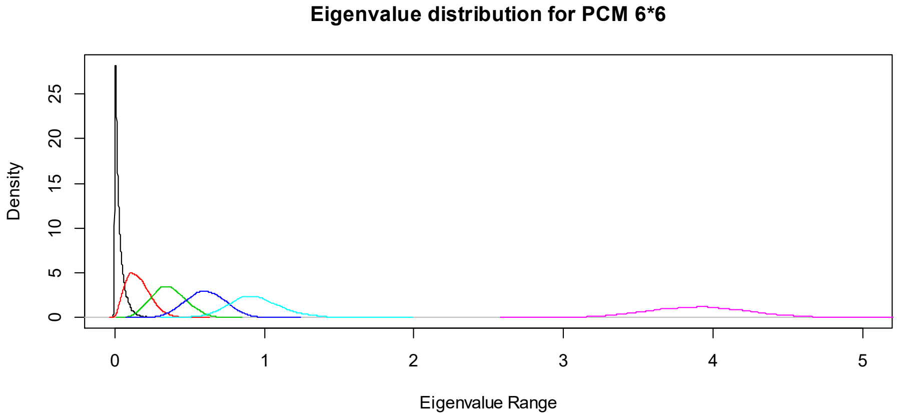
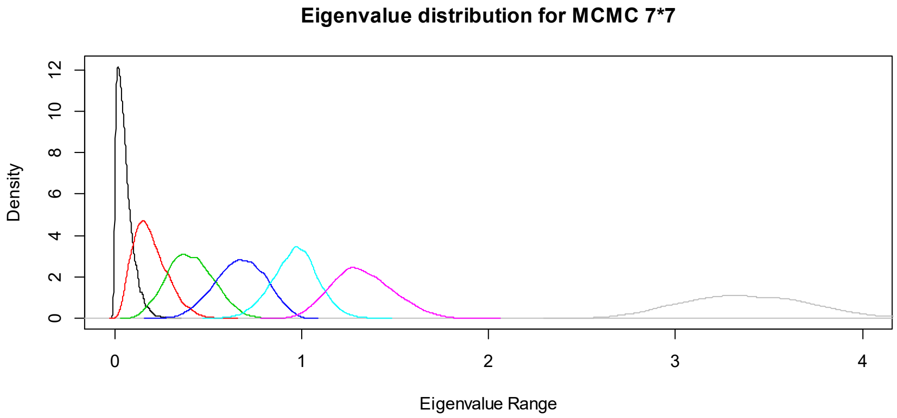
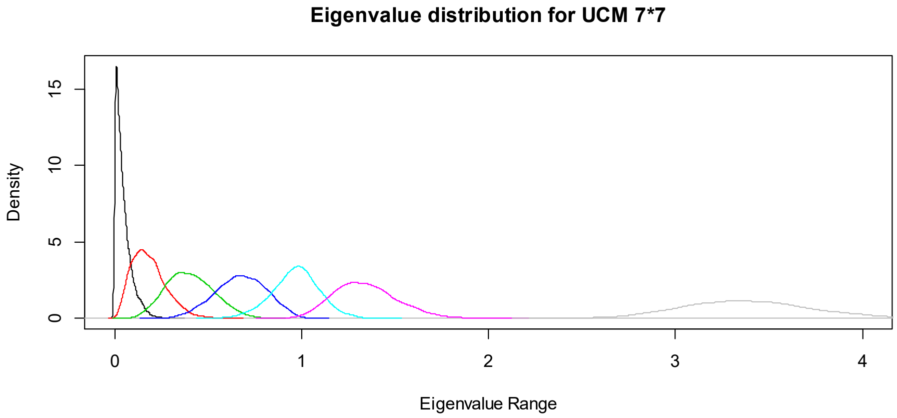


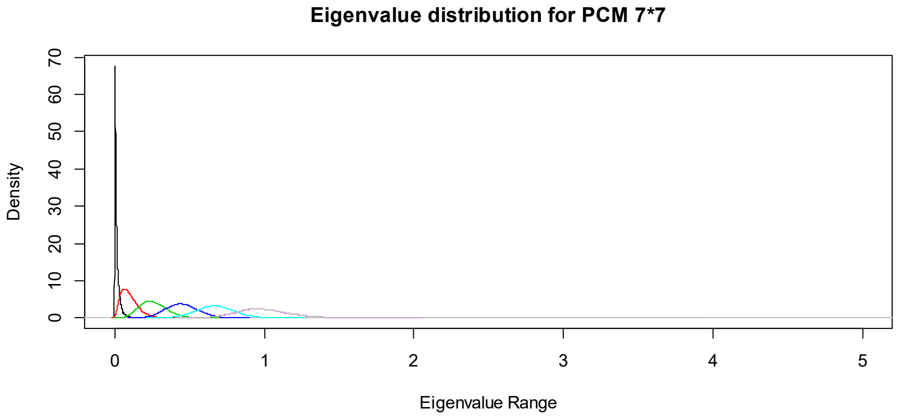
Figure 2.3.12: Eigenvalue distribution from UCM, MCMC, and PCM algorithms for 7 x 7 matrices.



Eigenvalue distribution for MCMC 7*7



Eigenvalue distribution for UCM 7*7



Eigenvalue distribution for PCM 7*7

In summary, UCM method is the simplest way to generate random uniform correlation matrices, when dimension of the matrices is less than 7. From our experience, based on 1,000 trials the empirical rejection rates are 0.584, 0.874, 0.978 and 0.998, respectively, for dimension 4, 5, 6 and 7. To generate 10,000 random uniform correlation matrices, the UCM method needs 6 seconds when the dimension is 4, 19 seconds when the dimension is 5, 2 minutes when the dimension is 6 and 30 minutes when the dimension is 7.

MCMC method is more complex than UCM method in coding and runs slower than UCM method when dimension is less than 7 because the MCMC method needs a large number of matrices to burn in. However, the speed of MCMC method is still acceptable when the dimension is small and when the dimension is larger than 6 it runs much faster than the UCM method. From our experience, to generate 10,000 matrices, MCMC method need 2 minutes 45 seconds when dimension is 4, 2 minutes 53 seconds when dimension is 5, 4 minutes when dimension is 6 and 4 minutes 36 seconds when dimension is 7.

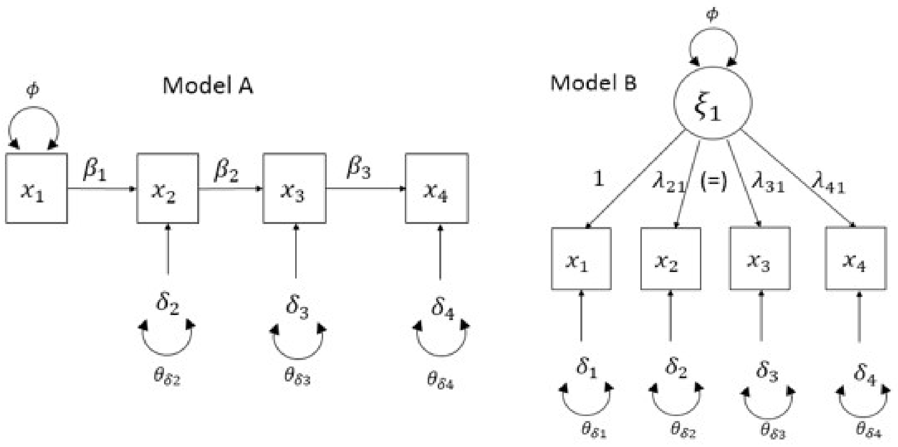
PCM method is the fastest way to generate positive definite correlation matrices because no generated matrices are rejected. It only needs 4 seconds to generate 10,000 4 by 4 matrices, 6 seconds for 5 by 5 matrices, 9 seconds for 6 by 6 matrices and 12 seconds for 7 by 7 matrices. But the matrices yielded from this method did not similar to the matrices yielded from MCMC or UCM.

In this study, random uniform correlation matrices with larger than 7 dimensions will be used to measure the model complexity, larger sample space is needed. So MCMC method was used to generate all the uniform correlation matrices.

2.4 A Two-Step Strategy for Model Complexity Comparison

Chapter 2 introduced two kinds of random correlation matrices to compare the model complexity between SEM models. The uniform random correlation matrices can be seen as the total data space for SEM models, by fitting SEM models to uniform random correlation matrices, we can access the complexity of competing SEM models and determine which model is more complex if we can observe a significant difference in their fitting results. For example, in the comparison between model A and Model B showed in Figure 2.4.1, 10,000 uniform random correlation matrices were generated using MCMC method with random seed 123.

Figure 2.4.1 Two Models with 4 observed variables, same model degrees of freedom but different functional forms

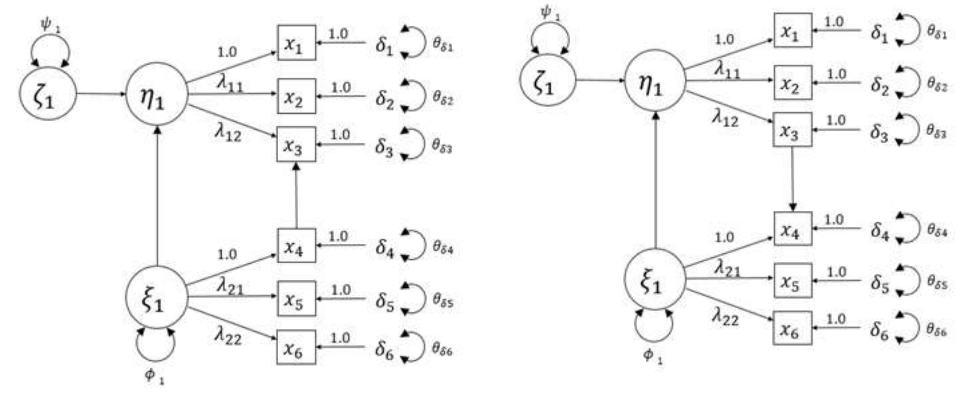


Setting the sample size equal to 100 and fitting all uniform random correlation matrices to both Model A and Model B, Model A fits well (p-value of chi-square test larger than 0.05) in N_A cases, Model B fits well in N_B cases and they both fit well in N_{AB} cases. Thus, we can determine the complexity of Model A as it fits well in N_A out of 10,000 random samples and the complexity of Model B as it fits well in N_B out of 10,000 random samples. If $N_B > N_A$ we can say Model B fits better than Model A in uniform random correlation matrices, therefore Model B is more complex than Model A. Otherwise, when $N_B < N_A$ Model A is more complex than Model B.

When the number of observed variables is small (4 observed variables in this case), the probability of a model fitting well in uniform random correlation matrices for a particular SEM model is high (near 3% for Model A and near 9% for Model B). Thus, in these cases, the model complexity can be determined in a reasonable time and we can decide which model is more complex since they have very different model complexity (in this case, R 3.3.1 only needs 12 minutes to get fitting results of one SEM model in 10000 uniform random correlation matrices). However, when the number of observed variables increased, the model fitting process will take more time (i.e. R 3.3.1 needs 2 hours to get fitting results of one SEM model in 10,000 6*6 uniform random correlation matrices). Also, the SEM models will have lower probability of fitting well in uniform random correlation matrices. Data generation settings like the random seed will affect the model complexity result, therefore larger random data sample will be needed to determine model complexity. Moreover, when the model complexity of SEM models are very close (as often happens when models have same model degrees of freedom), the uniform random correlation matrices will not be suitable to assess the model complexity comparison.

For example, in the comparison of Model A and Model B in Figure 2.4.2, two models both have 6 observed variables.

Figure 2.4.2 Two Models with 6 observed variables, same model degrees of freedom but different functional forms



Model A – Path model with x_4 regressed on x_3, q=14, t=6

Model B – Path model with x_3 regressed on x_4, q=14, t=6

We generated 10,000 uniform random correlation matrices with random seed 123, set the sample size equal to 100 and fit all data sets to both Model A and Model B. R 3.3.1 took 4 hours to get all the fitting results. From these results, the frequencies of data sets fitted well by Model A or Model B are both low (< 100), therefore, using different random seed will affect the results of complexity comparison. In this kind of cases, we tried to increase the number of uniform random correlation to get better comparison results.

Thus, we fitted Model A and Model B again using 100,000 uniform random correlation matrices with random seed 123. This time, even though the frequencies of data sets fitted well by Model A or Model Bare about 10 times higher (Model A fitted well in 595 random samples, Model B fitted well in 629 random samples and they both fitted well in 417 random samples), it is still hard to say which model is more complex, even though we spent 40 hours fitting the models.

In these kind of cases, we use known-model random population correlation matrices instead of uniform random correlation matrices to further examine SEM model complexity. In the comparison of Model A and Model B in Figure 3.3.4, first we used know-model data generation method in 2.1.2 to generate 10,000 random population correlation matrices with random seed 123 from Model A, in these random samples, Model A fits perfectly (p-value=1.00) in 10,000 cases, and Model B fits well in 7,306 cases. Then we used the same method to get 10,000 random samples from Model B, Model B fits perfectly in 10,000 cases and Model A fits well in 7,635 cases. Thus, say Model B is more complex than Model A.

In this case, from the fitting results of SEM models in uniform random correlation matrices, we can see both Model A and Model B have low probability of fitting well (both lower than 1%) and the model complexity between these two models are very close. Thus, known-model random population correlation matrices are needed to do the comparison of model complexity.

In conclusion, in this dissertation, on the purpose to examine how tetrads can be used to access SEM model complexity, determine the model complexity (especially for models have same degrees of freedom) is a crucial part. Both uniform random correlation matrices and known-model random population correlation matrices should be used. For every comparison of SEM models in this study, 10,000 uniform random correlation matrices will be generated and used to get a general estimate of model complexity for models in the comparison first, then for each model in the comparison, 10000 known-model random population correlation matrices should be generated and used to compare the complexity of SEM models. Sample size will be set to 100 for every SEM models in comparison.

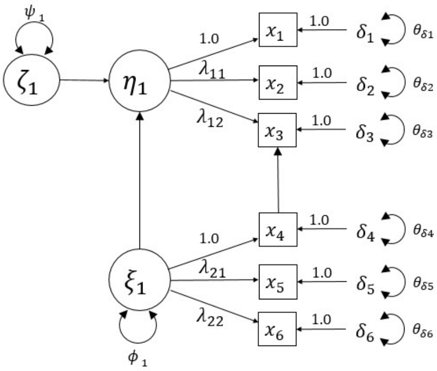
2.5 R codes for Tetrad Analysis

In order to determine the tetrad numbers of SEM models, a function named "TetradAnalysis" in the R system for statistical computing was created and published on GitHub (https://github.com/Hangcheng1989/TetradAnalysis). It takes input of a covariance matrix, a specified SEM model, and a sample size and returns the test statistic for the multivariate tetrad test, p-value for multivariate tetrad test, a set of nonredundant vanishing tetrads implied by the inputted SEM model, and the number of nonredundant vanishing tetrads implied by inputted SEM model.

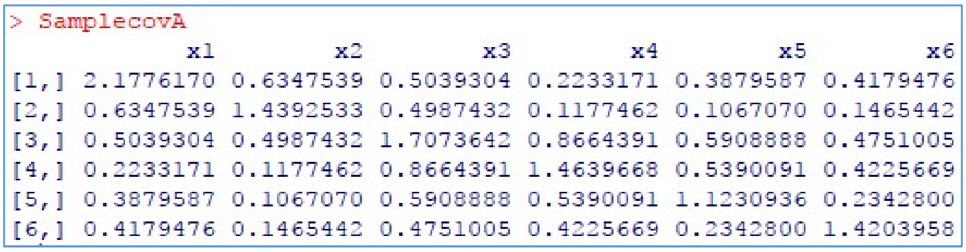
The inputted covariance matrix must have a minimum of four dimensions because the tetrad analysis requires at least four variables in the model. The column names of the inputted covariance matrix must match the names used in the model specification of inputted SEM model. In this tetrad analysis function, the R package lavaan (Rosseel 2012) was used to produce the model implied covariance matrix, thus the specification of the inputted model must follow the model specification rules of lavaan.

Below is an example showing the procedure for using the created r function, "TetradAnalysis." In this example, the tested model is a SEM model with two latent variables ($\eta_1$and ξ_1) and six observed variables ($x_1$to x_6), each latent variable is related to three observed variables and variable x_4 is also a indicator of variable x_3 (see Figure 2.5.1).

Figure 2.5.1 CFA model with 6 observed variables and two latent variables

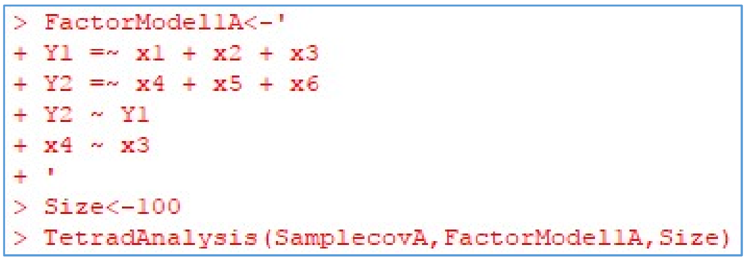


A sample covariance matrix “SamplecovA” shown below is generated from this model with sample size equal to100.



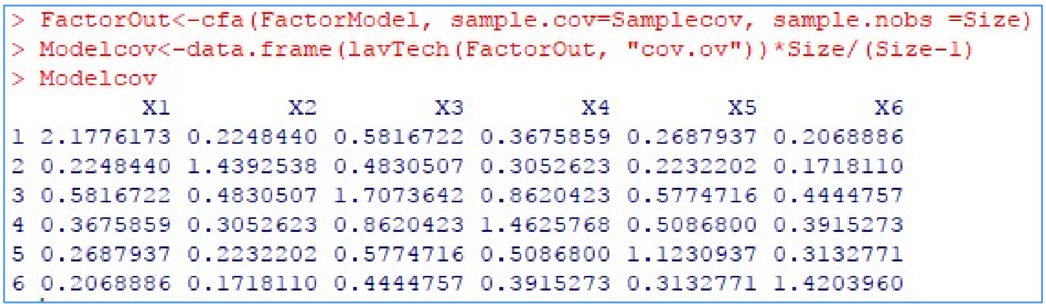
```
> SamplecovA
            x1        x2        x3        x4        x5        x6
[1,] 2.1776170 0.6347539 0.5039304 0.2233171 0.3879587 0.4179476
[2,] 0.6347539 1.4392533 0.4987432 0.1177462 0.1067070 0.1465442
[3,] 0.5039304 0.4987432 1.7073642 0.8664391 0.5908888 0.4751005
[4,] 0.2233171 0.1177462 0.8664391 1.4639668 0.5390091 0.4225669
[5,] 0.3879587 0.1067070 0.5908888 0.5390091 1.1230936 0.2342800
[6,] 0.4179476 0.1465442 0.4751005 0.4225669 0.2342800 1.4203958
```

The following R code shows the specification of inputted model and how to use function “TetradAnalysis”. The syntax used here is the same as syntax in package “lavaan” (Rosseel, 2012).



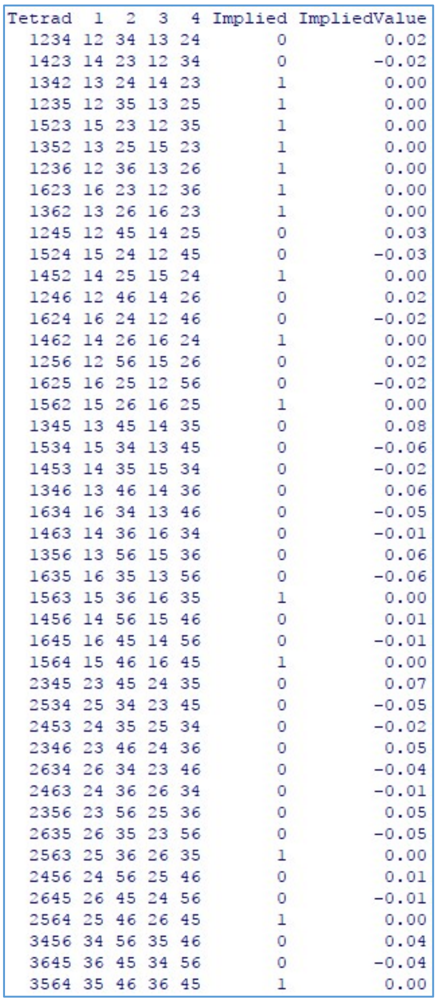
```
> FactorModel1A<-'
+ Y1 =~ x1 + x2 + x3
+ Y2 =~ x4 + x5 + x6
+ Y2 ~ Y1
+ x4 ~ x3
+ '
> Size<-100
> TetradAnalysis(SamplecovA,FactorModel1A,Size)
```

In the tetrad analysis process, first we use the “cfa” function in lavaan to do the model analysis, then we can got the model implied covariance matrix named “Modelcov” from the result of “cfa” function.



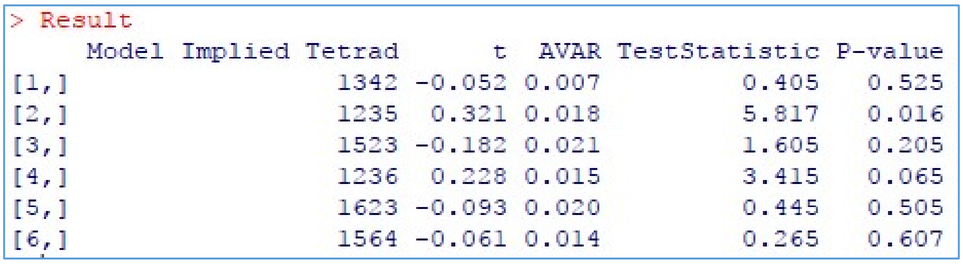
```
> FactorOut<-cfa(FactorModel, sample.cov=Samplecov, sample.nobs =Size)
> Modelcov<-data.frame(lavTech(FactorOut, "cov.ov"))*Size/(Size-1)
> Modelcov
         X1        X2        X3        X4        X5        X6
1 2.1776173 0.2248440 0.5816722 0.3675859 0.2687937 0.2068886
2 0.2248440 1.4392538 0.4830507 0.3052623 0.2232202 0.1718110
3 0.5816722 0.4830507 1.7073642 0.8620423 0.5774716 0.4444757
4 0.3675859 0.3052623 0.8620423 1.4625768 0.5086800 0.3915273
5 0.2687937 0.2232202 0.5774716 0.5086800 1.1230937 0.3132771
6 0.2068886 0.1718110 0.4444757 0.3915273 0.3132771 1.4203960
```

Then using the model implied covariances, the empirical method from Bollen (1993) can be used to identify which tetrads were vanished in the inputted SEM model. “TetradAnalysis” can provide us a list of the tetrads implied by inputted model and which are vanishing in the empirical method. The first column showed the tetrad labels, column 2 to 5 showed the covariances included in each tetrad setting, column showed the indicate the tetrad is vanishing or not (1=yes, 0=no) and the last column showed the tetrad residual value calculated from the model implied covariances. If the absolute value of a tetrad residual is larger than 0.001, this tetrad is a vanishing tetrad in the inputted model.



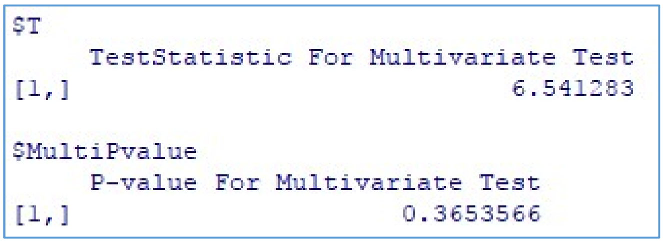
```
Tetrad  1  2  3  4 Implied ImpliedValue
  1234 12 34 13 24       0         0.02
  1423 14 23 12 34       0        -0.02
  1342 13 24 14 23       1         0.00
  1235 12 35 13 25       1         0.00
  1523 15 23 12 35       1         0.00
  1352 13 25 15 23       1         0.00
  1236 12 36 13 26       1         0.00
  1623 16 23 12 36       1         0.00
  1362 13 26 16 23       1         0.00
  1245 12 45 14 25       0         0.03
  1524 15 24 12 45       0        -0.03
  1452 14 25 15 24       1         0.00
  1246 12 46 14 26       0         0.02
  1624 16 24 12 46       0        -0.02
  1462 14 26 16 24       1         0.00
  1256 12 56 15 26       0         0.02
  1625 16 25 12 56       0        -0.02
  1562 15 26 16 25       1         0.00
  1345 13 45 14 35       0         0.08
  1534 15 34 13 45       0        -0.06
  1453 14 35 15 34       0        -0.02
  1346 13 46 14 36       0         0.06
  1634 16 34 13 46       0        -0.05
  1463 14 36 16 34       0        -0.01
  1356 13 56 15 36       0         0.06
  1635 16 35 13 56       0        -0.06
  1563 15 36 16 35       1         0.00
  1456 14 56 15 46       0         0.01
  1645 16 45 14 56       0        -0.01
  1564 15 46 16 45       1         0.00
  2345 23 45 24 35       0         0.07
  2534 25 34 23 45       0        -0.05
  2453 24 35 25 34       0        -0.02
  2346 23 46 24 36       0         0.05
  2634 26 34 23 46       0        -0.04
  2463 24 36 26 34       0        -0.01
  2356 23 56 25 36       0         0.05
  2635 26 35 23 56       0        -0.05
  2563 25 36 26 35       1         0.00
  2456 24 56 25 46       0         0.01
  2645 26 45 24 56       0        -0.01
  2564 25 46 26 45       1         0.00
  3456 34 56 35 46       0         0.04
  3645 36 45 34 56       0        -0.04
  3564 35 46 36 45       1         0.00
```

After the model implied vanishing tetrads were determined, the sweep operator was used to identify sets of nonredundant vanishing tetrads. Thus, we can also provide the number of nonredundant vanishing tetrads of inputted model, use the sample covariance matrix to conduct the individual tetrad tests for each nonredundant vanishing tetrad and the multivariate tetrad test for the whole model. The first column of list “Result” showed there were 6 nonredundant vanishing tetrads implied by the model, the second model provides the nonredundant vanishing tetrads, the third column showed the sample value for each tetrad, the forth column showed asymptotic variance for the given tetrad, the fifth column showed the test statistic for each individual tetrad test, and the p-value showed in last column can be used to determine whether the tetrad is vanishing.



```
> Result
     Model Implied Tetrad      t  AVAR TestStatistic P-value
[1,]                 1342 -0.052 0.007         0.405   0.525
[2,]                 1235  0.321 0.018         5.817   0.016
[3,]                 1523 -0.182 0.021         1.605   0.205
[4,]                 1236  0.228 0.015         3.415   0.065
[5,]                 1623 -0.093 0.020         0.445   0.505
[6,]                 1564 -0.061 0.014         0.265   0.607
```

The final section of the output showed the test statistic and the p-value for multivariate tetrad test. In this case, the p-value is larger than 0.05, which means the inputted model is consistent with the sample data.



```
$T
     TestStatistic For Multivariate Test
[1,]                            6.541283

$MultiPvalue
     P-value For Multivariate Test
[1,]                     0.3653566
```

2.6 Investigating the Relationship between Model Complexity and Tetrad Numbers

2.6.1 The Strategy to Determine SEM Model Complexity

Two kinds of random correlation matrices were introduced in this chapter, the uniform random correlation matrices and known-model random correlation matrices. The uniform random correlation matrices represent the data space relevant to SEM models uniformly. In this study, one criterion of model complexity is defined as how well the SEM models can fit the uniform random correlation matrices and the known-model random correlation matrices. In general, a more complex model will have higher a convergence rate when applied to both uniform random correlation matrices and known-model random correlation matrices. Furthermore, in uniform random correlation matrices where both models can converge, the more complex model will have better fitting results in most fit indices.

When comparing models, the number of data sets that converge by models is the first criterion for comparison, model with higher convergence rate in uniform random correlation matrices should be more complex. For the random correlation matrices where both converge, model complexity can be further determined by comparing models' overall performance measured by many different fit indices. Absolute fit indices RMSEA, RMR and SRMR, incremental fit indices TLI, CFI and IFI, and parsimony fit indices AIC and BIC will be used in this part. After fitting models to the same random correlation matrices, the complexity of competing SEM models can be determined by comparing the mean value of fitting results, the frequencies of datasets fit well by models (number of data sets that models have better results than the threshold value for a fit index, i.e. SRMR <0.08) and the frequencies of data sets one model fits better than the other (i.e. number of data sets that one model has lower SRMR value than the other). All acceptable threshold levels for fit indices mentioned in this section are listed in Table

1.2.1 and Table 1.2.2. Density plots and cumulative frequency distributions of fitting results from those fit indices are also be provided.

After the definition of strategy to determine model complexity, the selection of models that will be useful to determine the situations where vanishing tetrad number can be useful to indicate model complexity is another important part of this study. The rules for model selection are described in following sections.

2.6.2 Models with Same Number of Free Parameters but Different Vanishing Tetrads

To test if the model complexity is related to the number of model implied vanishing tetrads, several model pairs that have same model degrees of freedom but different vanishing tetrad numbers will be compared. It is expected that when applying these models to same random data, their goodness of fit will not be the same. Additionally, it is expected that when SEM models have same model degrees of freedom, the model that has larger vanishing tetrad number is the more complex model. That is, the model with the larger vanishing tetrad numbers will fit better in both uniform random correlation matrices and known model random correlation matrices.

2.6.3 Models with Same Number of Free Parameters, Same Vanishing Tetrads but Different Constraint Types

Vanishing tetrad number cannot replace the role of model degrees of freedom to indicate model complexity, it should be a complementary of model degrees of freedom. Because in many situations, when some constraints (i.e. set constant parameter values) are added to a SEM model, neither tetrad number nor number of free parameters will change. This means some model complexity change will not be detected by vanishing tetrad numbers, thus vanishing tetrad number cannot be used to indicate model complexity alone. To demonstrate this, comparisons between models with same degrees of freedom, same vanishing tetrads but different constraint types will be used.

2.6.4 Summary of Comparison Settings

There will be two types of comparisons between different SEM models in this study. In each comparison, all models will base on the same random sample (random correlation matrices), which means they will have same number of observed variables. In Section 2.6.2, SEM models in one comparison will have same model degrees of freedom, and different vanishing tetrad numbers. Based on the results from these comparisons, we can determine the relationship between model complexity and number of vanishing tetrads.

In Section 2.6.3, SEM models in one comparison will have same vanishing tetrad number and same model degrees of freedom. Models may have the same functional form but different parameter settings. Thus, the results of comparison are expected to show that SEM model complexity cannot be fully assessed even model degrees of freedom and vanishing tetrad number are both used.

Based on the comparisons in Section 2.6.2 and 2.6.3, our hypothesis that vanishing tetrad should be a complement of model degrees of freedom to indicate SEM model complexity will be proved.

All the data generation processes, SEM model building, the values of fit indices and the comparison of overall model fit will be conducted in R 3.3.1. For the data generation part, three sets of R commands will be created to perform UCM, MCMC, partial correlation and known model random data generation and for each comparison, the most suitable data generation method will be used. For the vanishing tetrad analysis, an R 3.3.1 package will be written to calculate all the model implied tetrads, determine which of them are vanished, and also select a set of nonredundant vanishing tetrads from the vanishing tetrads. Thus, the vanishing tetrad number of the SEM models in our study can be calculated very easily. Value of fit indices in this study will be calculated using R package “lavaan”.

Chapter 3 Results

3.1 Overview of Chapter 3

Section 1.4 introduced Confirmatory Tetrad Analysis (CTA) as a complement of traditional SEM methodologies to test model fit because testing vanishing tetrads provide a goodness-of-fit test for a model that can lead to results different from the usual likelihood-ratio test associated with maximum likelihood (ML)/ weighted least squares (WLS) methods. Additionally, in some model comparison situations, models are not nested in traditional likelihood-ratio (LR) test are nested in terms of vanishing tetrads, hence a nested test in CTA can be used for model selection in this situation. When two models with nested tetrads have close fitting results, the model with fewer vanishing tetrads is preferred, just like nested model comparisons in the traditional LR test, model where a fewer number of unknown parameters is preferred. From Chapter 1, we already know that in traditional SEM fit indices the number of free parameters is often used to indicate model complexity, and we also find evidence that shows only using the number of free parameters is not always appropriate for evaluating model complexity. Since CTA can be seen as a complement of traditional SEM methodologies to test model fit, we want to know if the number of vanishing tetrads can also be used as a complement of number of free parameters to indicate SEM model complexity. The goal of this chapter will be to explore if the number of vanishing tetrads can be used to indicate SEM model complexity, especially when models have same number of free parameters.

In this chapter, several model testing scenarios are proposed. Structural equation models with same number of free parameters but different functional forms will be fit to the generated random data. Using the two-step model complexity comparison strategy mentioned in section 2.4, the degree of complexity of particular models can be assessed by fitting uniform random correlation matrices and the relative model complexity to other models in the same comparison setting can be assessed by fitting known-model random population correlation matrices. Each model is fit to 10,000 uniform random correlation matrices generated by the MCMC method mentioned in section 2.3, then fit to 10,000 random population correlation matrices generated from itself and 10,000 random population correlation matrices generated from comparable model. By comparing the frequencies of data sets fit well by models in comparison and mean value of commonly used SEM fit indices, model complexity can be assessed. Comparison results are summarized in tables and plots. After model complexity for each model was accessed, the R function mentioned in section 2.5 was be used to conduct Confirmatory Tetrad Analysis (CTA), thus identifying number of vanishing tetrads. From these results, we can demonstrate that in SEM models, functional form can contribute to model complexity and it may also contribute to the number of vanishing tetrads.

3.2 Complexity Due to Functional Forms: Different Relationship between Two Variables

When considering SEM models, functional forms refers to the set of simultaneous equations relating observed variances and covariances to free parameters (Preacher 2006). In SEM models, to explain the relationship between two related variables, two different relationships are often used, one is the effect of one variable on another, represented by a direct effect in path diagram, and the other one is correlation/covariance between variables, represented by a double-headed curved arrow. Both effect and correlation/covariance relationships add one unknown parameter to the model, which means that when using the number of unknown parameters to evaluate model complexity, they are the same. However, adding an effect or correlation/covariance relationship in SEM model will have different influences on model implied vanishing tetrad numbers in CTA, thus two models shown in Figure 3.2.1 and 3.2.2 were specified to examine how this small change in functional form will affect the model complexity.

Model 1A and 1B are both based on 6 observed variables x_1 to x_6. The exogenous latent variable ξ_1 has an effect on endogenous latent variable η_1. A direct effect was used between x_3 and x_4, in Model 1A, and in Model 1B, we assume the random error terms of x_3 and x_4 are correlated.

Figure 3.2.1 Model 1A – Path model with x_4 regressed on x_3

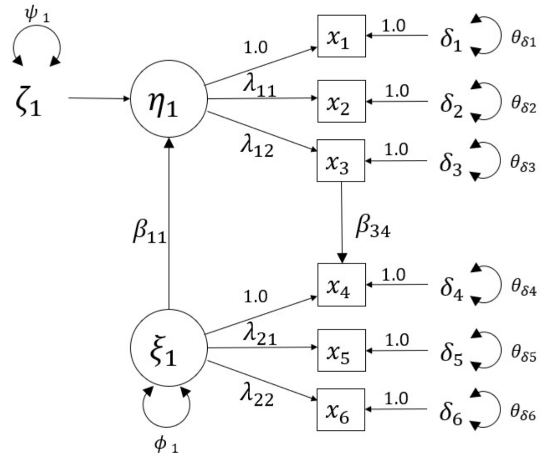
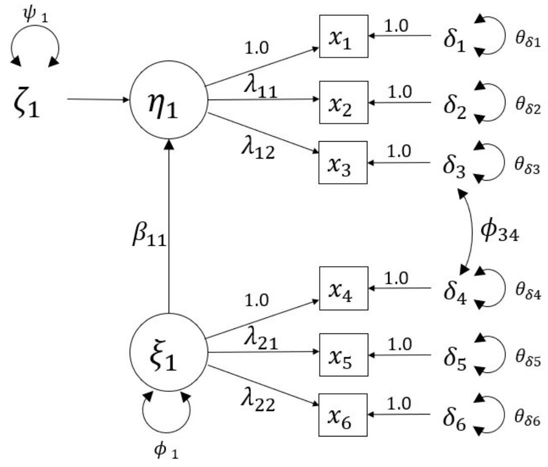


Figure 3.2.2 Model 1B – Path model with the random error terms of x_4 and x_3 correlated



After we fit these two models to uniform random correlation matrices, the convergence rate for Model 1A was 75.26% (7,526 out of 10,000 matrices) while the convergence rate for Model 1B is 79.34% (7,934 out of 10,000 matrices). The two models both converged in 6,877 matrices and their average fitting results are listed in Table 4.2.1, histograms and cumulative frequency distribution (CDF) plots of fitting results are showed in Figure 3.2.3.

Table 3.2.1. Fitting results for Model 1A and 1B in 6,877 uniform random correlation matrices

	P-value of χ^2 test		**SRMR**		**RMSEA**		**GFI**	
Threshold Level	> 0.05		< 0.08		< 0.1		> 0.95	
	1A	**1B**	**1A**	**1B**	**1A**	**1B**	**1A**	**1B**
Mean	0.002	0.002	0.152	0.160	0.361	0.358	0.814	0.816
SD	0.032	0.032	0.077	0.089	0.130	0.126	0.063	0.063
N of fit well	62	57	536	461	60	56	111	106
N of fit better	3333	3544	3969	2908	3329	3542	3294	3583

	CFI		**TLI**		**AIC**		**BIC**	
Threshold Level	> 0.95		> 0.95					
	1A	**1B**	**1A**	**1B**	**1A**	**1B**	**1A**	**1B**
Mean	0.741	0.745	0.446	0.454	1436.605	1434.747	1473.077	1471.219
SD	0.130	0.130	0.281	0.277	105.040	106.713	105.040	106.713
N of fit well	186	178	186	178				
N of fit better	3329	3542	3333	3544	3333	3544	3333	3544

Figure 3.2.3. Histograms and CDF plots of fitting results for Model 1A (Red) and 1B (Blue) to 6,877 uniform random correlation matrices

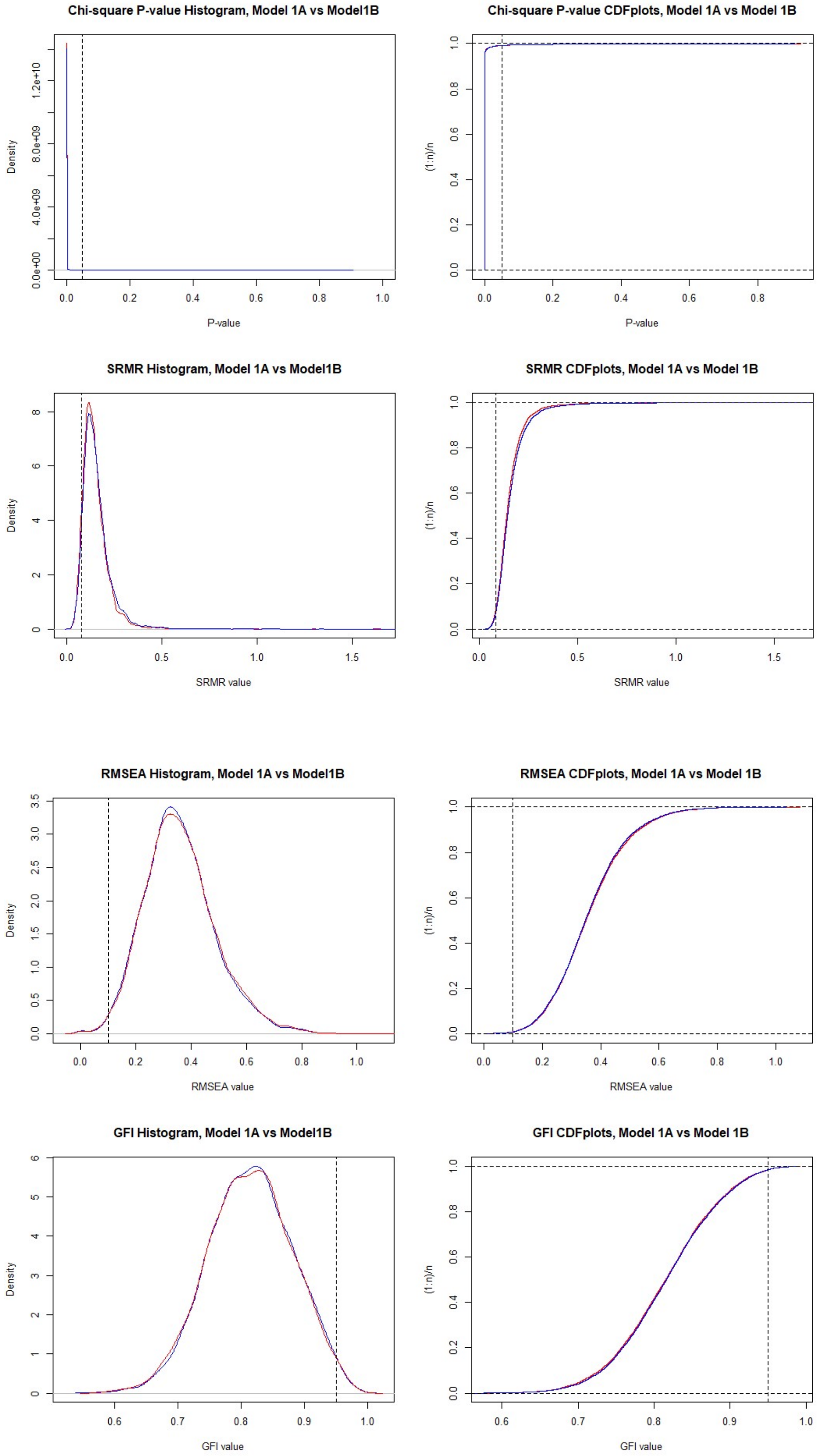
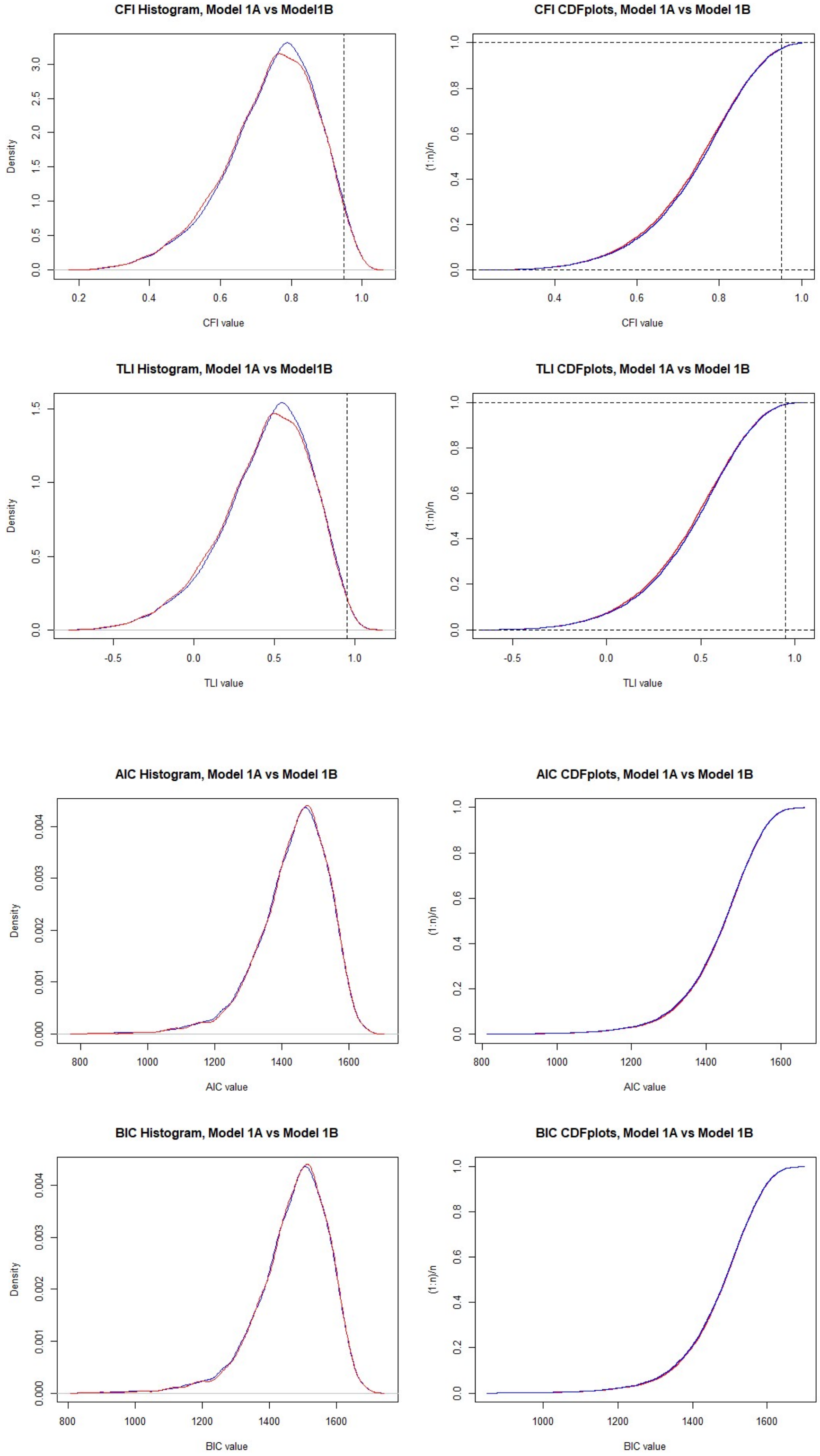


Figure 3.2.3. Histograms and CDF plots of fitting results for Model 1A (Red) and 1B (Blue) to 6,877 uniform random correlation matrices



From the fitting results for those random datasets where Model 1A and 1B both converged, Model 1B has better mean values than 1A in most fit indices (SRMR, GFI, CFI, TLI, AIC, BIC), has better fitting results in more data sets using most fit indices (χ^2, RMSEA, GFI, CFI, TLI, AIC, BIC). Histograms and CDF plots confirmed these results are consistent. Although the frequency of data sets fitted well by Model 1A is slightly higher than Model 1Busing most fit indices, considering Model 1B has much higher convergence rate (79.34% vs 75.26%), Model 1B fits better than Model 1A in uniform random correlation matrices.

Known-model random population correlation matrices were also used to ensure we obtain the correct comparison results in the previous comparison. In random population correlation matrices generated from Model 1A, using chi-square test as a fit index, Model 1B fits well (p-value > 0.05) in 98.34% (9,834 out of 10,000) cases. In correlation matrices generated from Model 1B, Model 1A fits well in 82.84% (8,284 out of 10,000) cases. Hence this comparison result can confirm Model 1B is more complex than Model 1A.

As expected, Model 1B fits random data better than Model 1A, although they have same number of unknown parameters, which means a correlation/covariance relationship is more complex than a direct effect relationship and number of unknown parameter (or degrees of freedom) cannot recognize this difference. Then the R function "TetradAnalysis" was used to conduct Confirmatory Tetrad Analysis for both Model 1A and 1B. From the CTA results, 6 vanishing tetrads ($\tau_{1342}, \tau_{1235}, \tau_{1523}, \tau_{1236}, \tau_{1623}, \tau_{1564}$) are implied by Model 1A while 7 vanishing tetrads ($\tau_{1342}, \tau_{1235}, \tau_{1523}, \tau_{1236}, \tau_{1623}, \tau_{1564}, \tau_{1645}$) are implied by Model 1B, moreover, Model 1A is nested in Model 1B in terms of vanishing tetrad. Thus, the nested tetrad test can be used in this case to know which model fits better in confirmatory tetrad analysis. Their fitting results in tetrad analysis for uniform random correlation matrices are listed in Table 3.2.2. In Confirmatory

Tetrad Analysis, because Model 1A and 1B are nested in terms of tetrads and 1B has more vanishing tetrads, Model 1A should be preferred if the result from nested tetrad test is significant (p-value < 0.05). From CTA results, although 1A has a higher frequency of data sets fitted well (1,649 data sets for 1A and 1,384 data sets for 1B), nested tetrad test should be used as the gold standard to identify which model fits better. From nested tetrad test results, 1A was chosen as a better model in 19.54% uniform random correlation matrices (1,954 significant better results found from 10,000 samples). Therefore, nested tetrad test results from CTA is another evidence to show Model 1B fits better in random data and in this comparison, model with more vanishing tetrad is more complex.

Table 3.2.2. Fitting results from CTA for Model 1A and 1B in 10,000 uniform random correlation matrices

	P-value of tetrad test		**P-value of nested tetrad test**
Threshold Level	> 0.05		**< 0.05**
	1A	**1B**	**1A nested in 1B**
Mean	0.039	0.031	0.413
SD	0.102	0.086	0.370
N of fit well	1649	1384	1954

3.3 Complexity Due to Functional Forms: Different Levels of Specification

When SEM models have different functional forms, one model may have specified stronger assumptions than another even though they have same number of unknown parameters. This will make this model more difficult to fit random data. For example, Model 2A in Figure 3.3.1 is a simple model based on 6 observed variables x_1 to x_6 with two correlated latent variables. Model 2B in Figure 3.3.2 based on same data, but Model 2B assumes two latent variables are uncorrelated, and use an extra factor loading of x_3 on latent variable ξ_1 instead. So these two models have same number of degrees of freedom (*df* = 8) but different functional form. Clearly Model 2B is more specific than Model 2A since many variables are not allowed to be correlated in 2B while in Model 2A all variables are correlated. This specification will make model more difficult to fit random data because the values in observed random correlation matrices are unlikely to be zero.

Figure 3.3.1 Model 2A – A simple CFA with two factors correlated

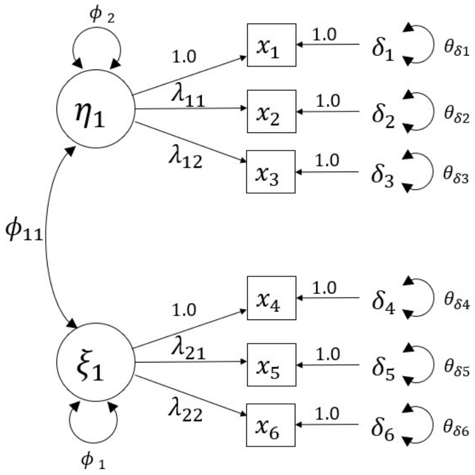
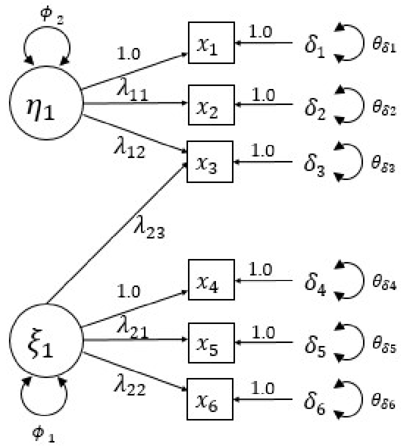


Figure 3.3.2 Model 2B – A simple CFA with x_3 regressed on factor ξ_1



When fitting to uniform random correlation matrices, Model 2A converged in 8,163 matrices while Model 2B converged 5,131 matrices, both converged in 4,372 matrices. The two models have very different convergence rates because Model 2B specified an extra factor loading for x_3 instead of using a correlation/covariance constraint between two latent variables. The fitting results for Model 2A and 2B in those both converged matrices are listed in Table 3.3.1, histograms and cumulative frequency distribution (CDF) plots of the fitting results are showed in Figure 3.3.3.

Table 3.3.1. Fitting results for Model 2A and 2B in 4,327 uniform random correlation matrices.

	P-value of χ^2 test		**SRMR**		**RMSEA**		**GFI**	
Threshold Level	> 0.05		< 0.08		< 0.1		> 0.95	
	2A	**2B**	**2A**	**2B**	**2A**	**2B**	**2A**	**2B**
Mean	0.001	0.000	0.164	0.335	0.377	0.453	0.786	0.745
SD	0.000	0.000	0.095	0.321	0.130	0.130	0.071	0.071
N of fit well	21	0	214	0	23	1	30	1
N of fit better	1527	230	4105	267	3520	852	3453	919
	CFI		**TLI**		**AIC**		**BIC**	
Threshold Level	> 0.95		> 0.95					
	2A	**2B**	**2A**	**2B**	**2A**	**2B**	**2A**	**2B**
Mean	0.691	0.555	0.420	0.166	1450.268	1500.534	1484.135	1534.402
SD	0.141	0.164	0.265	0.307	100.806	99.397	100.806	99.397
N of fit well	44	5	44	5				
N of fit better	3520	852	3520	852	3520	852	3520	852

Figure 3.3.3. Histograms and CDF plots of fitting results for Model 2A (Red) and 2B (Blue) to 4,327 uniform random correlation matrices

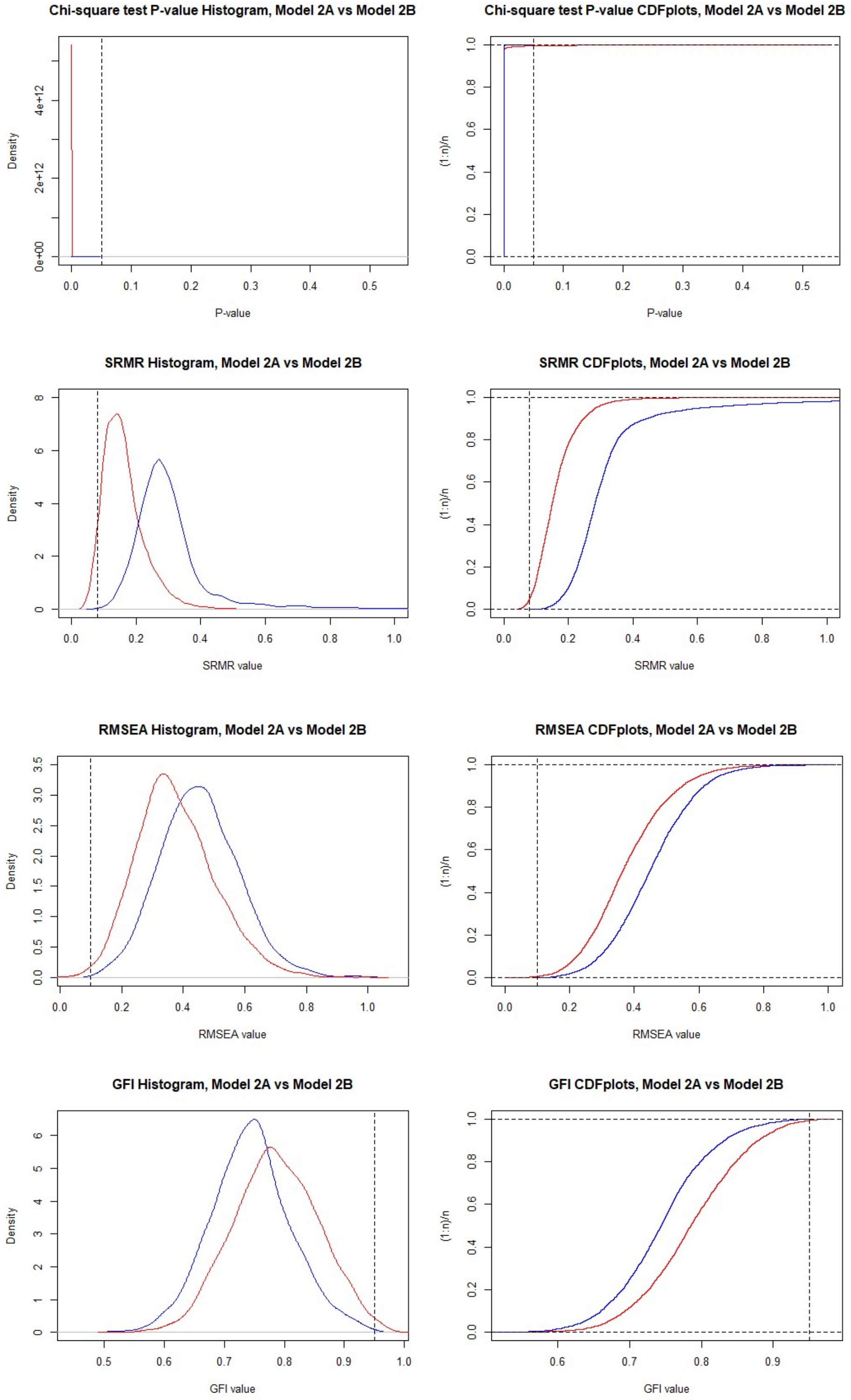
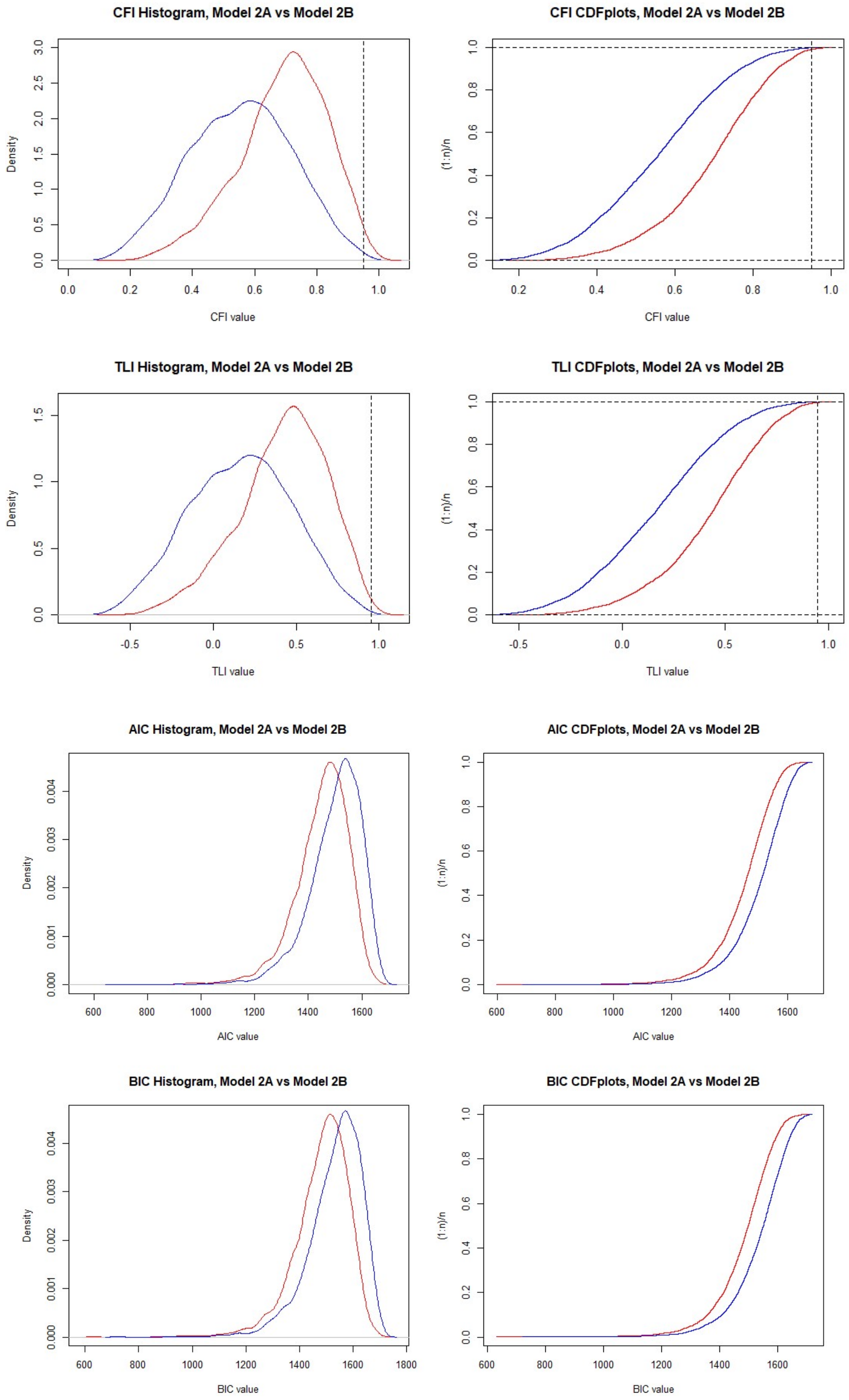


Figure 3.3.3. Histograms and CDF plots of fitting results for Model 2A (Red) and 2B (Blue) to 4,327 uniform random correlation matrices



From fitting results for those 4,327 uniform random correlation matrices Model 2A and 2B both converged, Model 2A has better fit than 2B in all fit indices showed in Table 3.3.1, when comparing mean values, frequencies of data sets fit well and frequencies of data sets fit better. Histograms and CDF plots confirmed these results are consistent. In random population correlation matrices generated from Model 2A, using the chi-square test as a fit index, Model 2A fits perfectly in all cases and Model 2B only fits well in 2.08% (208 out of 10,000) cases. In correlation matrices generated from Model 2B, Model 2B fits perfectly in all cases and Model 2A fits well in 28.74% (2,874 out of 10,000) cases. These results proved Model 2A is more complex than Model 2B when their model degrees of freedom are both equal to 8.

Although these two models have same model degrees of freedom, they have different numbers of vanishing tetrads. Seven vanishing tetrads were found in Model 2B (τ_{3564}, τ_{3645}, τ_{2564}, τ_{2645}, τ_{1564}, τ_{1645}, τ_{1362}) and 8 vanishing tetrad was found in Model 3A (τ_{3564}, τ_{3645}, τ_{2564}, τ_{2645}, τ_{1564}, τ_{1645}, τ_{1362}, τ_{1623}), which means Model 2B is nested in Model 2A in terms of vanishing tetrads. Their fitting results in CTA for 10,000 uniform random correlation matrices are listed in Table 3.3.2.

Table 3.3.2. Fitting results from CTA for Model 2A and 2B in 10,000 uniform random correlation matrices

	P-value of tetrad test		**P-value of nested tetrad test**
Threshold Level	> 0.05		**< 0.05**
	2A	**2B**	**2B nested in 2A**
Mean	0.032	0.036	0.406
SD	0.080	0.090	0.305
N of fit well	1434	1622	1410

From CTA results, although Model 2A fit well (p-value of tetrad test >0.05) in fewer cases than 2B (1,434 for 2A and 1,622 for 2B), Model 2B is nested in 2A in terms of tetrads and nested tetrad test result shows Model 2B was selected as a better model in 14.10% data sets (1,410 significant better results found from 10,000 samples). Therefore, 2A is more complex than 2B from CTA results and it has more vanishing tetrads.

3.4 Complexity Due to Functional Form: Different Regions of Good Fit

When developing SEM models, different functional forms can be built on the same data and different functional forms are designed to fit different patterns of correlation matrix. Among them, some functional forms can fit more kinds of correlation patterns than others, thus it is more complex.

To illustrate this idea, two SEM models with very different functional forms are shown in Figure 3.4.1 and Figure 3.4.2. Model 3A and 3B both have 4 observed variables x_1 to x_4. Model 3A is an unrestricted simplex model, often used when observed variables are repeated measurements of a single variable or when correlation matrix has a band-diagonal pattern. Model 3B is a one-factor model with two factor loadings constrained to be equal, and it is expected to be more complex because it can potentially fit more correlation patterns. Each of these two models have same model degrees of freedom (*df* = 3).

Figure 3.4.1 Model 3A – An unrestricted simplex model

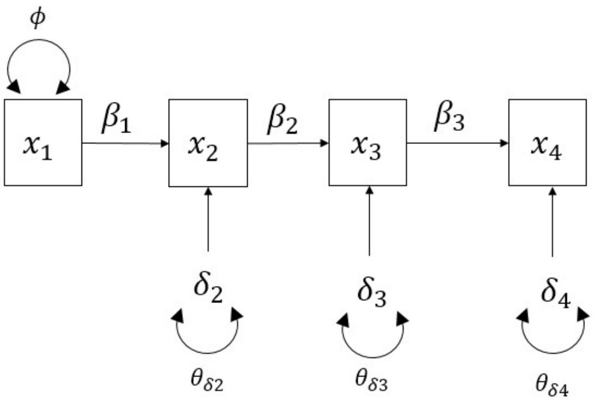
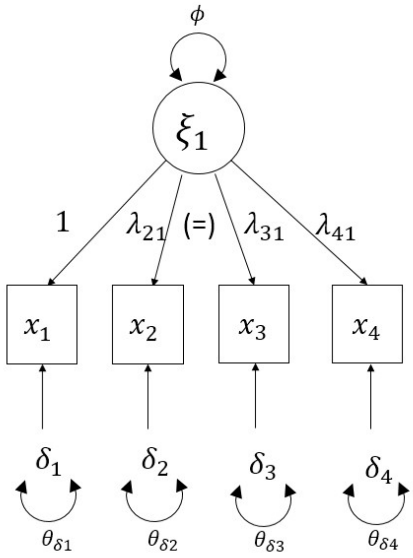


Figure 3.4.2 Model 3B – A one-factor model with one equality constraint



After fitting uniform random correlation matrices, Model 3A converged in all matrices while the convergence rate for Model 3B is 96.37% (9,637 out of 10,000). Fitting results for Model 3A and 3B in both converged matrices are listed in Table 4.4.1, histograms and cumulative frequency distribution (CDF) plots of fitting results are showed in Figure 3.3.3.

Table 3.4.1. Fitting results for Model 3A and 3B in 9,637 uniform random correlation matrices

	P-value of χ^2 test		**SRMR**		**RMSEA**		**GFI**	
Threshold Level	> 0.05		< 0.08		< 0.1		> 0.95	
	3A	**3B**	**3A**	**3B**	**3A**	**3B**	**3A**	**3B**
Mean	0.010	0.030	0.195	0.221	0.536	0.370	0.781	0.852
SD	0.071	0.118	0.084	0.255	0.247	0.202	0.095	0.084
N of fit well	310	840	823	1330	214	616	470	1304
N of fit better	1571	6150	3836	5801	1910	7716	1774	7863
	CFI		**TLI**		**AIC**		**BIC**	
Threshold Level	> 0.95		> 0.95					
	3A	**3B**	**3A**	**3B**	**3A**	**3B**	**3A**	**3B**
Mean	0.271	0.508	0.054	0.537	1032.778	983.53	1048.409	1001.766
SD	0.321	0.277	0.516	0.322	75.686	90.097	75.686	90.097
N of fit well	101	297	213	648				
N of fit better	1910	7716	1914	7723	2067	7570	2271	7366

Figure 3.4.3 Histograms and CDF plots of fitting results for Model 3A (Red) and 3B (Blue) to 9,637 uniform random correlation matrices

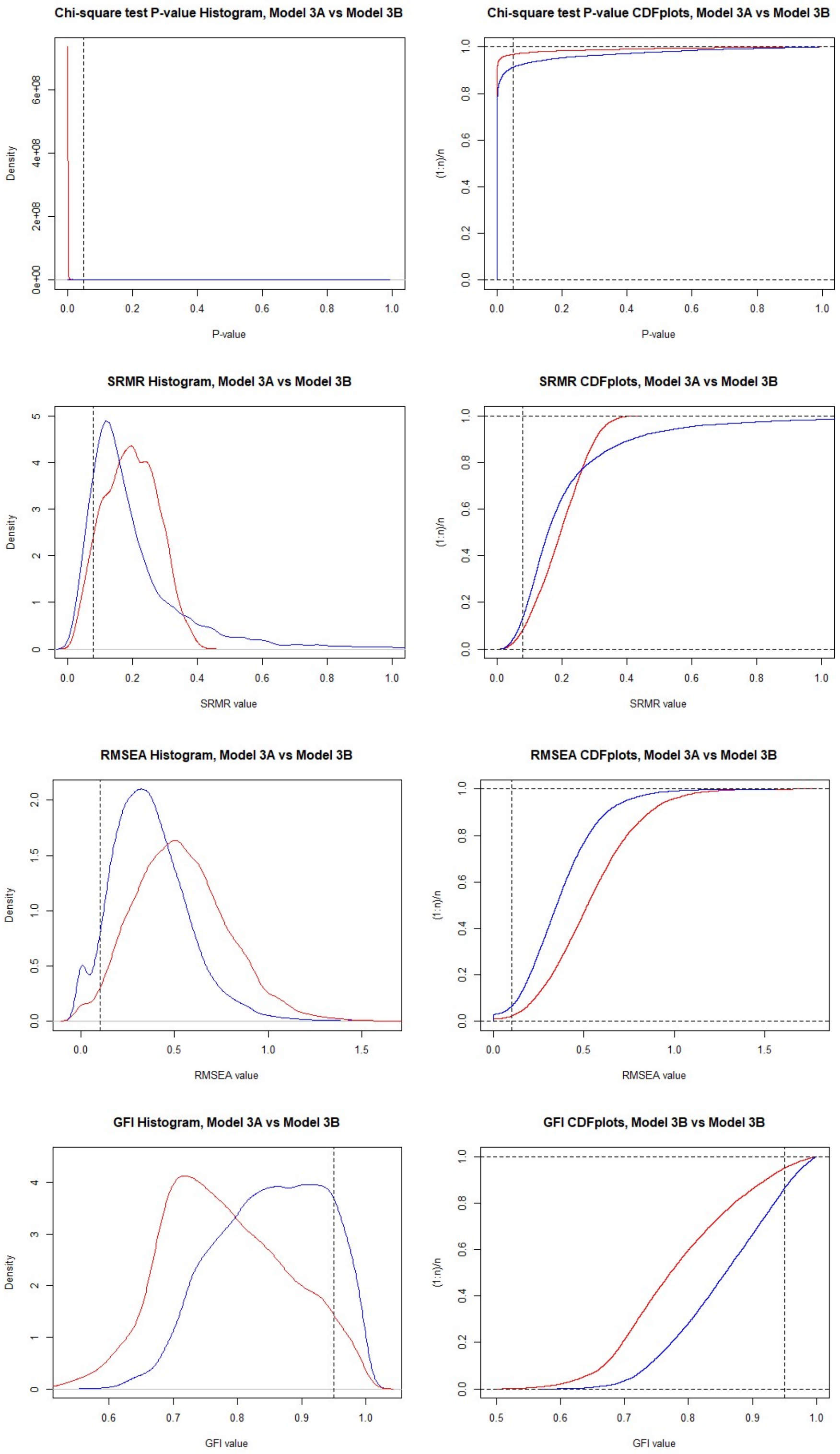
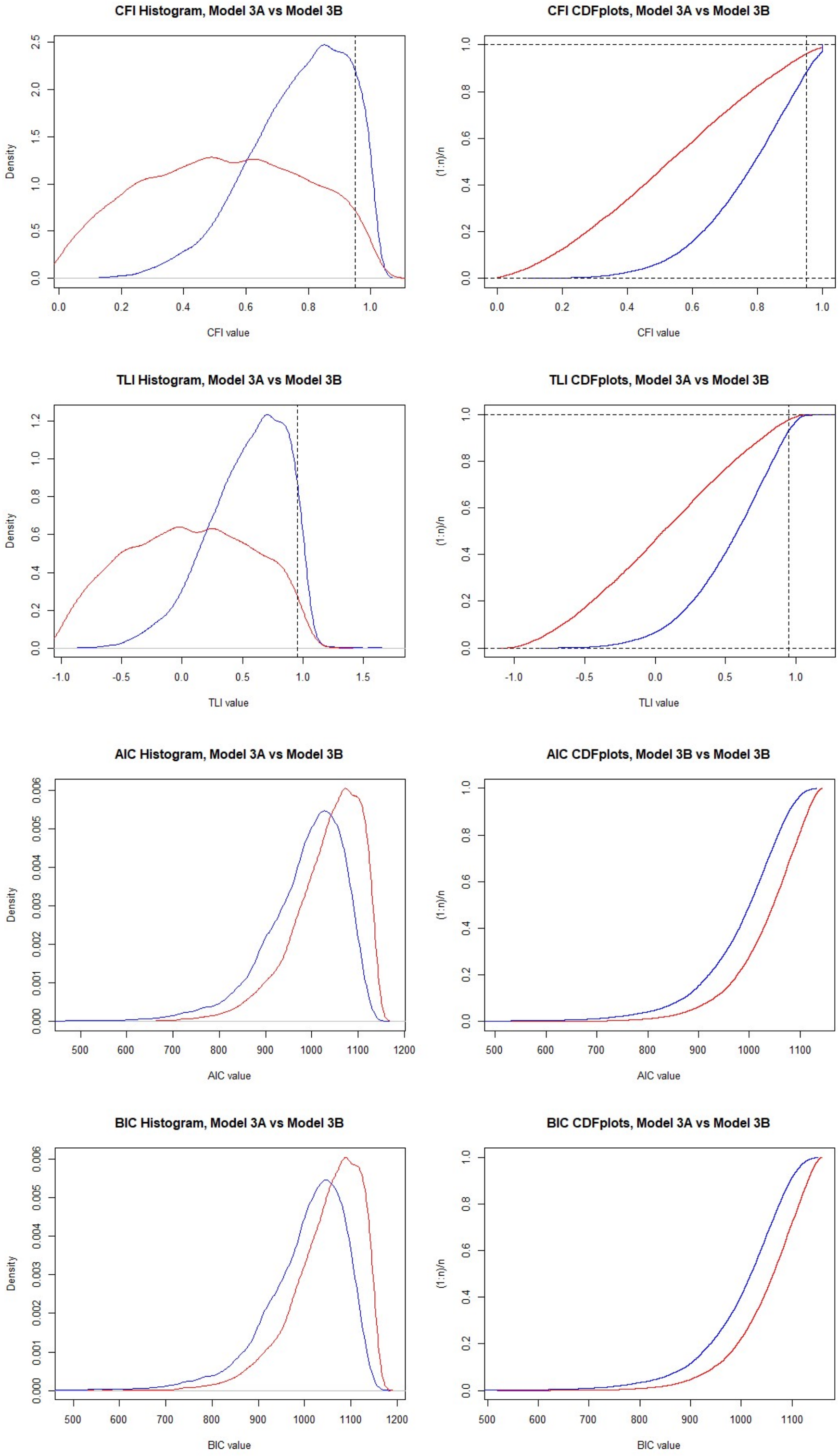


Figure 3.4.3 Histograms and CDF plots of fitting results for Model 3A (Red) and 3B (Blue) to 9,637 uniform random correlation matrices



From the fitting results of those 9,637 datasets Model 3A and 3B both converged, Model 3B has a higher frequency of data sets fit well and a higher frequency of data sets fit better in every fit index. Model 3B also has better mean values for most fit indices except for SRMR. Histograms and CDF plots confirm these results are consistent. Moreover, in 10,000 random population correlation matrices generated from Model 3A with random seed 123, using the chi-square test as a fit index, Model 3A fits well in all cases and Model 3B fits well in 2,942 cases. In 10,000 correlation matrices generated from Model 3B, Model 3B fits well in 10,000 cases and Model 3A fits well in 468 cases. These results show that Model 3B fits random data better than Model 3A, although they have same number of unknown parameters, which means models with different functional forms can be very different in model complexity.

In this comparison case, model implied vanishing tetrads for these two models can also tell the difference between the model complexity of the two models. Confirmatory Tetrad Analysis can recognize Model 3B as a more complex model because Model 3B implies 2 vanishing tetrads $(\tau_{1342}, \tau_{1234})$, and Model 3A has only one vanishing tetrad (τ_{1342}) which is nested in the vanishing tetrads implied by 3B. Their fitting results in CTA for uniform random correlation matrices are listed in Table 3.4.2.

Table 3.4.2. Fitting results from CTA for Model 3A and 3B in 10,000 uniform random correlation matrices

	P-value of tetrad test		**P-value of nested tetrad test**
Threshold Level	> 0.05		< 0.05
	3A	**3B**	**3A nested in 3B**
Mean	0.188	0.083	0.193
SD	0.277	0.187	0.278
N of fit well	4555	2446	9522

From CTA, Model 3A is nested in 3B and nested tetrad test result shows Model 3A was selected as a better model in 95.22% cases (9,522 significant better results found from 10,000 samples). That means based on nested tetrad test results in CTA, Model 3A should be a more complex model, however, all previous fitting result in this section against this idea thus we still conclude Model 3A is are more complex model and it has more vanishing tetrads.

Therefore, in the comparison between Model 3A and 3B, we further demonstrated the utility of using vanishing tetrad number as an index of SEM model complexity when two models have same number of degrees of freedom, different functional form and nested vanishing tetrad settings.

3.5 Complexity Due to Functional Form: Model Not Nested in Terms of Vanishing Tetrads

In section 3.2, 3.3 and 3.4, we demonstrated the feasibility of using the number of vanishing tetrads to compare model complexity when models are nested in terms of vanishing tetrads. In this section, we will examine if the vanishing tetrad number still has the ability to indicate model complexity even if two models are not nested in terms of vanishing tetrads.

Figure 3.5.1 Model 4A – Path model with x_3 as the central mediator

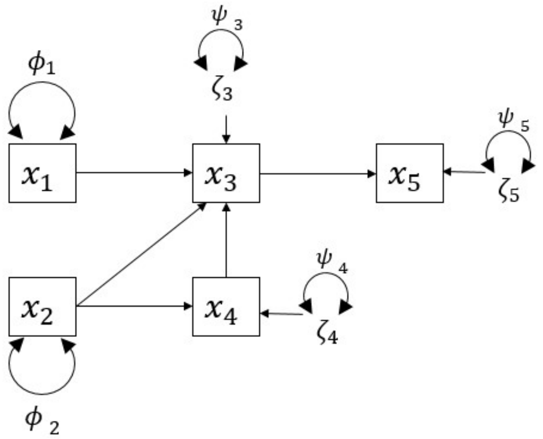
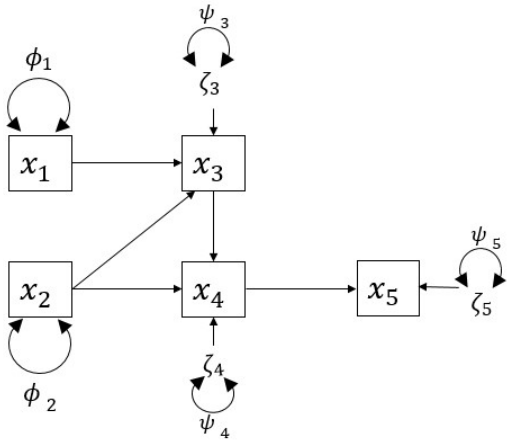


Figure 3.5.2 Model 4B – Path model with x_4 as the central mediator



In comparison 4, we have two CFA models (Model 4A in Figure 3.5.1 and Model 4B in Figure 3.5.2) based on 5 observed variables x_1 to x_5. In Model 4A shown in Figure 3.5.1, x_3 is used as the central mediator and it is directly connected to variable x_5. In Model 4B showed in Figure 3.5.2, x_4 is used as the central mediator. Two models have same degrees of freedom (*df* = 4) but different model implied vanishing tetrads. Model 4A has 3 vanishing tetrads(τ_{1234}, τ_{1235}, τ_{1345}), Model 4B has 2 vanishing tetrads(τ_{2134}, τ_{2135}) and they are not nested. The two step model complexity determination method was also used to compare their model complexity. This comparison allows us to examine if it is still true that model with more vanishing tetrads is more complex even two models are not nested in terms of tetrads.

When fit to uniform random correlation matrices, Model 4A and 4B both converged in all matrices. Fitting results for Model 4A and 4B in these cases are listed in Table 3.5.1, histograms and cumulative frequency distribution (CDF) plots of fitting results are shown in Figure 3.5.3.

Table 3.5.1 Fitting results for Model 4A and 4B in 10,000 uniform random correlation matrices.

	P-value of χ^2 test		**SRMR**		**RMSEA**		**GFI**	
Threshold Level	> 0.05		< 0.08		< 0.1		> 0.95	
	4A	**4B**	**4A**	**4B**	**4A**	**4B**	**4A**	**4B**
Mean	0.004	0.004	0.201	0.165	0.518	0.522	0.784	0.787
SD	0.002	0.002	0.201	0.063	0.207	0.209	0.084	0.080
N of fit well	138	138	619	939	101	96	264	262
N of fit better	2432	2295	3843	6157	5054	4942	4761	5239
	CFI		**TLI**		**AIC**		**BIC**	
Threshold Level	> 0.95		> 0.95					
	4A	**4B**	**4A**	**4B**	**4A**	**4B**	**4A**	**4B**
Mean	0.555	0.551	-0.001	-0.010	1247.010	1249.272	1267.851	1270.113
SD	0.222	0.219	0.500	0.492	93.059	88.875	93.059	88.875
N of fit well	216	194	216	194				
N of fit better	5054	4942	5056	4944	5056	4944	5056	4944

Figure 3.5.3 Histograms and CDF plots of fitting results for Model 4A (Red) and 4B (Blue) to 10,000 uniform random correlation matrices

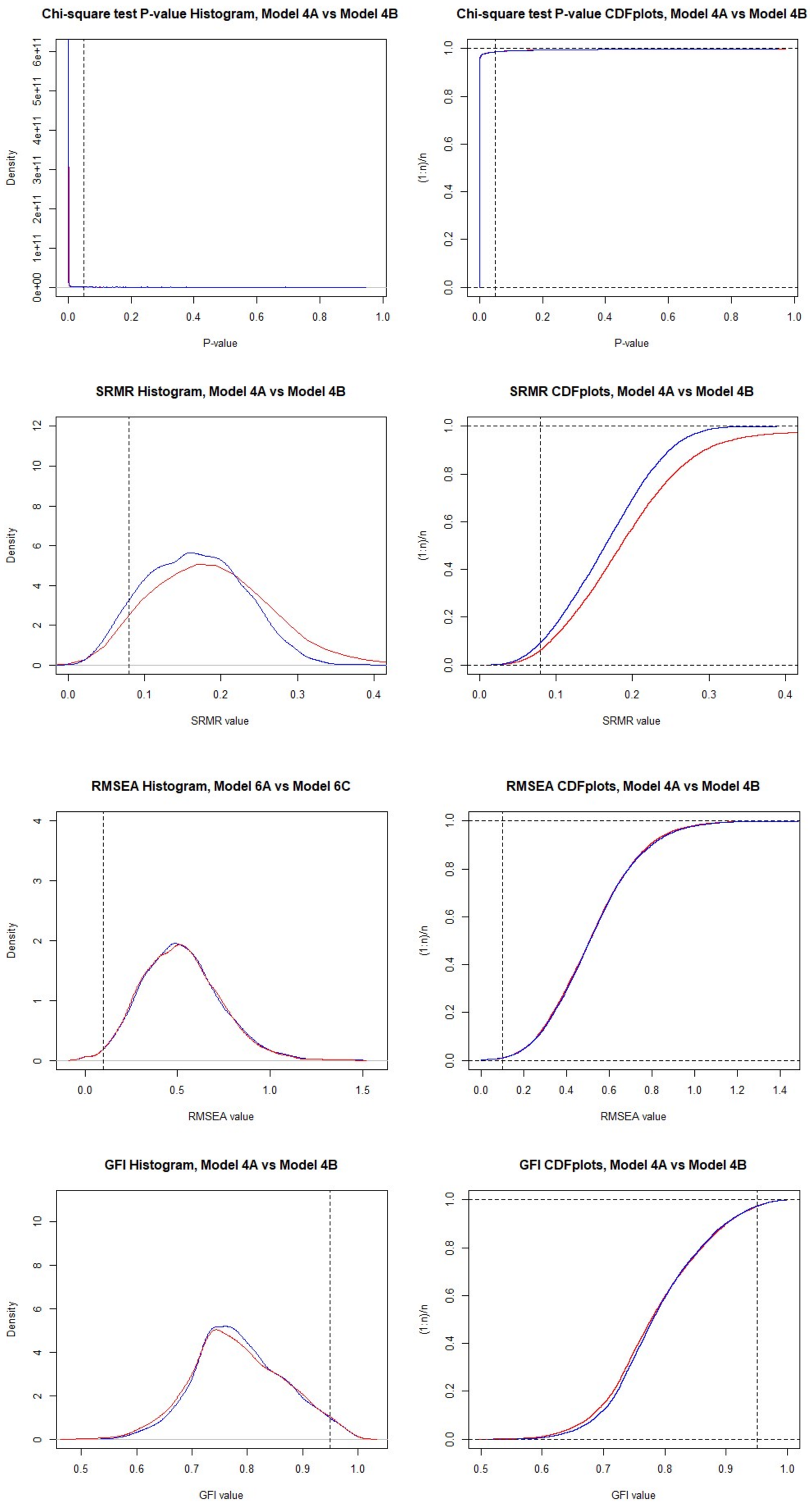
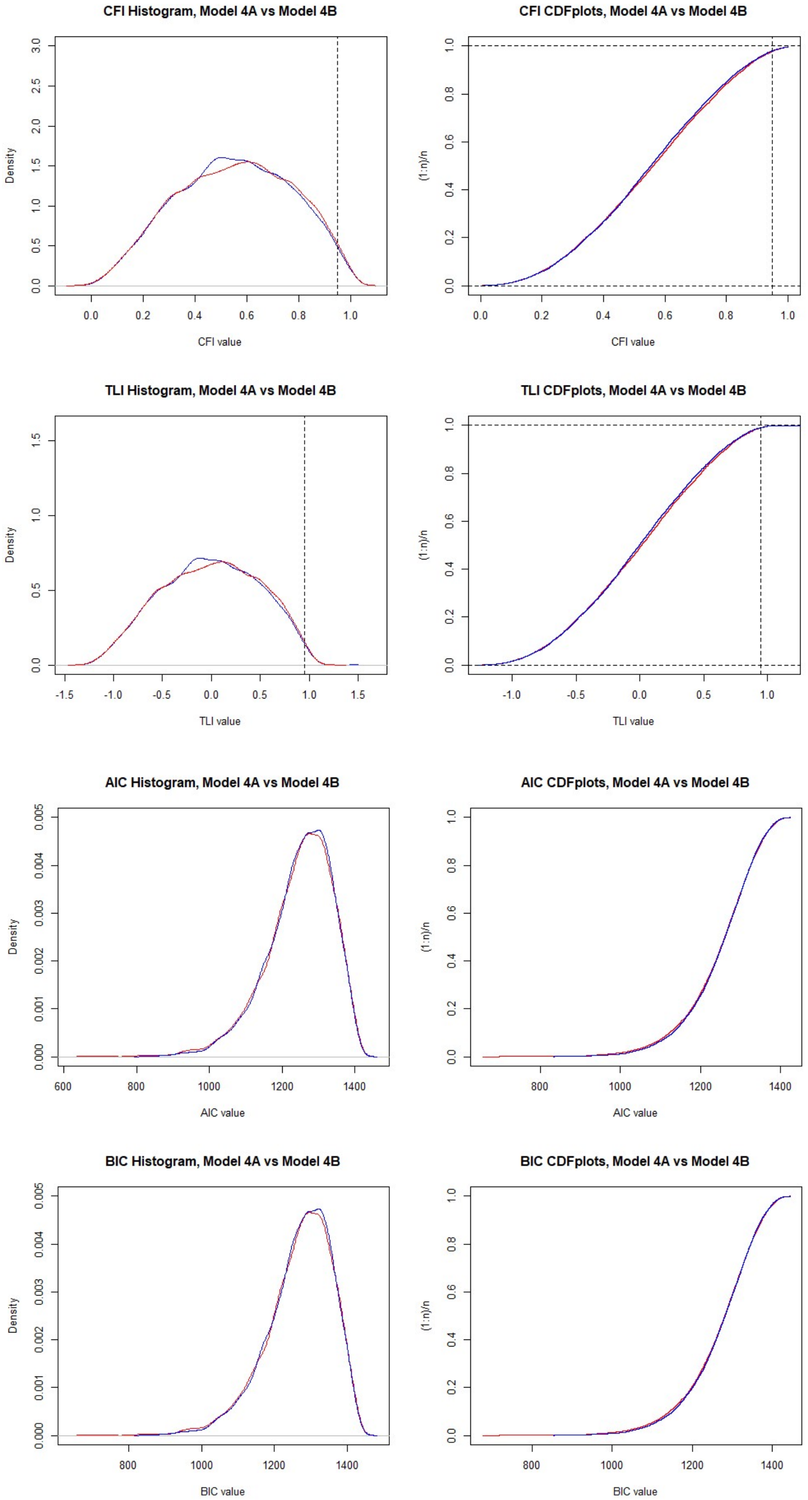


Figure 3.5.3 Histograms and CDF plots of fitting results for Model 4A (Red) and 4B (Blue) to 10,000 uniform random correlation matrices



From Table 3.5.1 and Figure 3.5.3, we can see Model 4A and 4B have very close fitting results in many fit indices, and Model 4A fits slightly better than 4B in most fit indices except SRMR. When examining their fitting results in known model analysis Model 4B can fit well (p-value of the chi-square test > 0.05) in 14.91% (1,491 out of 10,000) cases. Meanwhile Model 5B Model 4A can fit well in 21.34% (2,134 out of 10,000) cases.

Therefore, in our analysis Model 4A is a more complex model than Model 4B and in this comparison, vanishing tetrad number still works to indicate model complexity even though models in this comparison are not nested in terms of vanishing tetrads.

To further test the correctness of using vanishing tetrad number to indicate model complexity when models are not nested in terms of vanishing tetrads, one more comparison was tested in our study. In comparison setting 5, two models both have 9 observed variables x_1 to x_9. In Model 5A, x_3 was used as a mediator and Model 5B use x_1 and x_2 as two mediators to represent an alternative model building. Both models have 13 degrees of freedom but different model implied vanishing tetrad settings. Model 5A has 26 vanishing tetrads, Model 5B has 25 vanishing tetrads and they are not nested. Using uniform random correlation matrices and known model random population correlation matrices, we compared their model complexity.

Figure 3.5.4 Model 5A – Path model with x_3 as a mediator

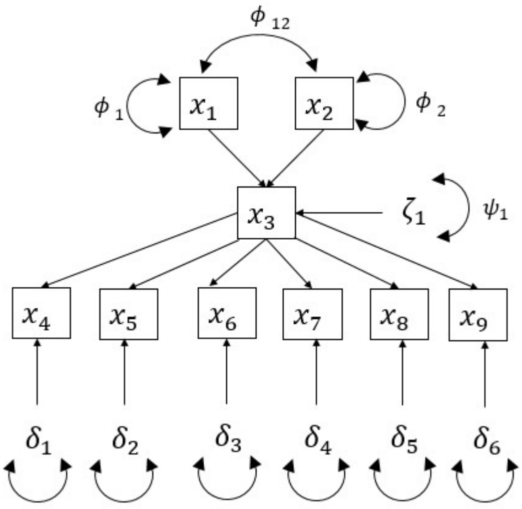
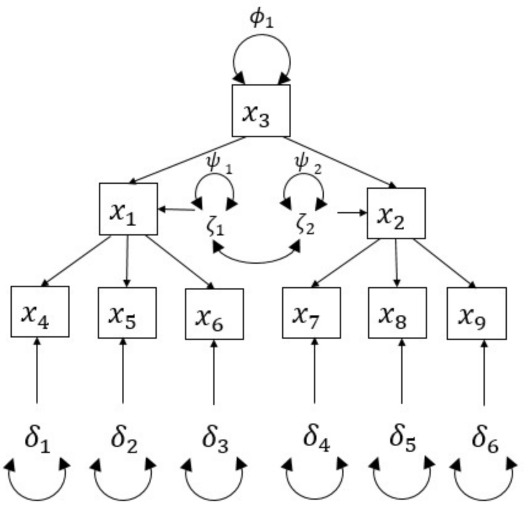


Figure 3.5.5 Model 5B – Path model with x_1 and x_2 as two mediators



When fit to uniform random correlation matrices, Model 5A converged in 81.63% (8,163 out of 10,000) matrices while Model 5B converged in all matrices. Fitting results for Model 5A and 5B in these both converged cases are listed in Table 3.5.2, histograms and cumulative frequency distribution (CDF) plots of fitting results are showed in Figure 3.5.6.

Table 3.5.2 Fitting results for Model 5A and 5B in 8,163 uniform random correlation matrices.

	P-value of χ^2 test		**SRMR**		**RMSEA**		**GFI**	
Threshold Level	> 0.05		< 0.08		< 0.1		> 0.95	
	5A	**5B**	**5A**	**5B**	**5A**	**5B**	**5A**	**5B**
Mean	0.000	0.000	0.158	0.193	0.450	0.436	0.786	0.779
SD	0.000	0.000	0.035	0.210	0.106	0.102	0.037	0.041
N of fit well	0	0	112	222	0	0	0	0
N of fit better	356	448	4727	5273	4113	5887	5851	4149
	CFI		**TLI**		**AIC**		**BIC**	
Threshold Level	> 0.95		> 0.95					
	5A	**5B**	**5A**	**5B**	**5A**	**5B**	**5A**	**5B**
Mean	0.375	0.416	-0.094	-0.023	2153.469	2136.744	2184.732	2168.006
SD	0.159	0.148	0.279	0.258	109.894	106.356	109.894	106.356
N of fit well	0	0	0	0				
N of fit better	4113	5887	4113	5887	4113	5887	4113	5887

Figure 3.5.6 Histograms and CDF plots of fitting results for Model 5A (Red) and 5B (Blue) to 8,163 uniform random correlation matrices

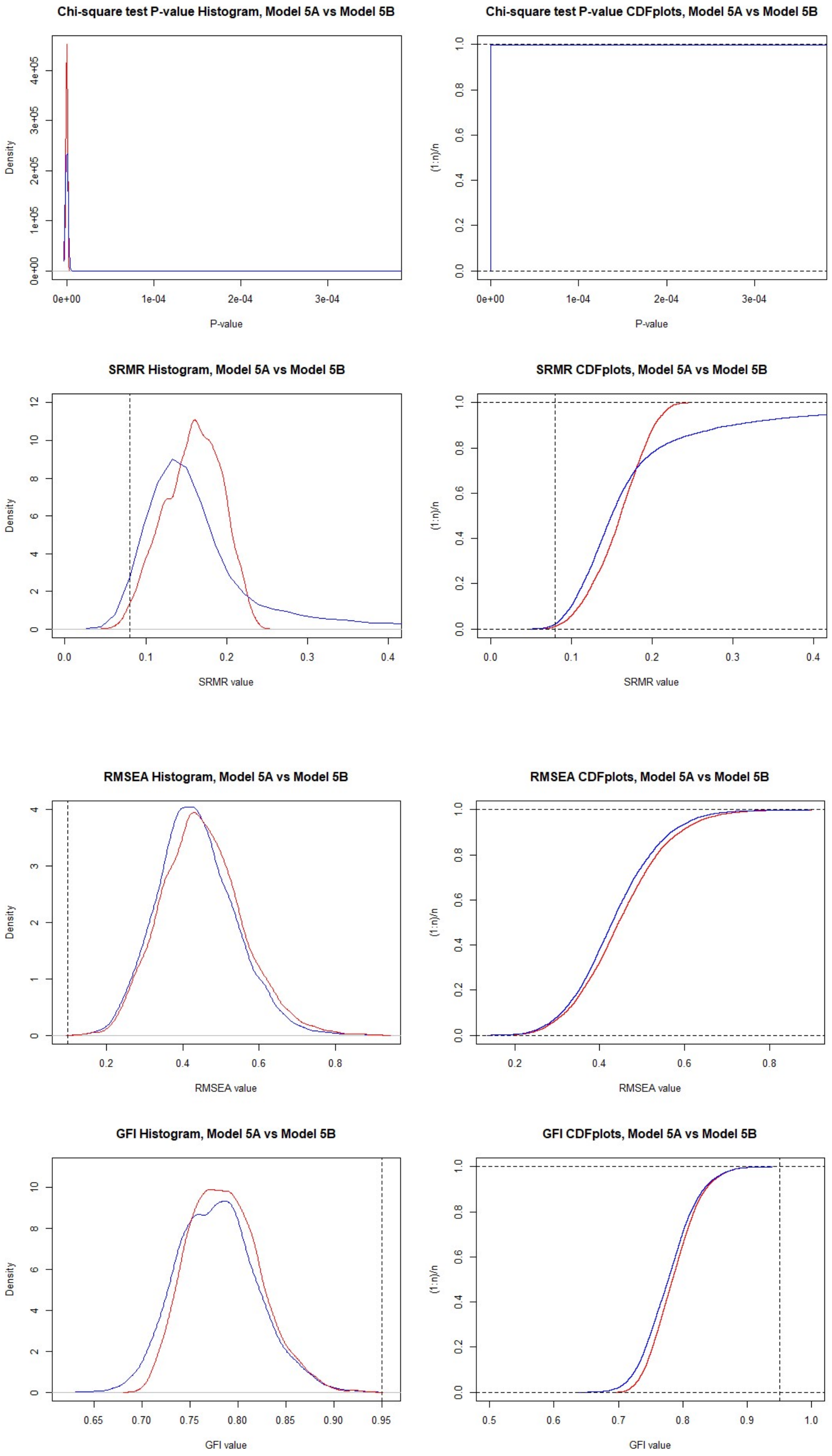
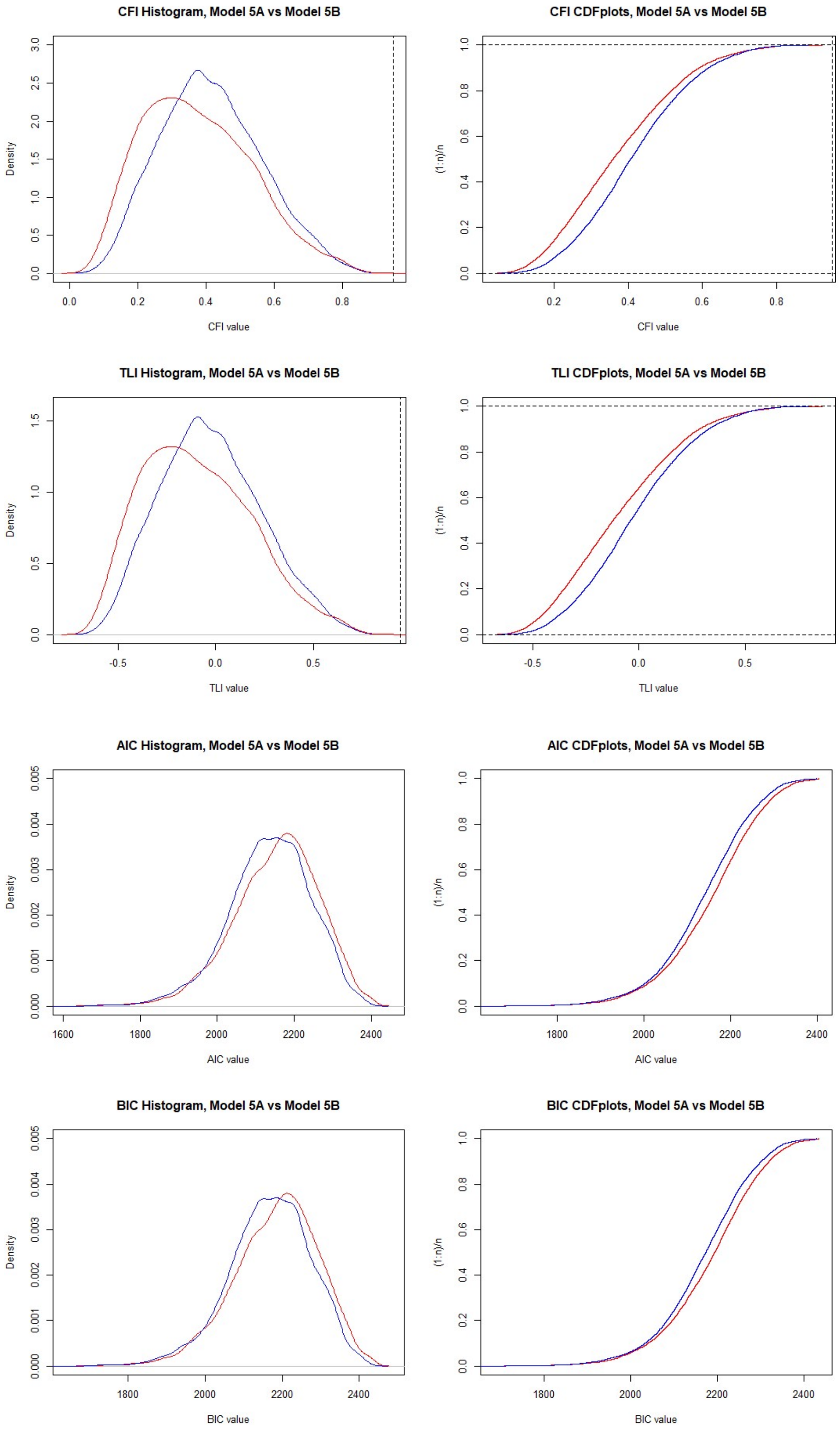


Figure 3.5.6 Histograms and CDF plots of fitting results for Model 5A (Red) and 5B (Blue) to 8,163 uniform random correlation matrices



From fitting results showed in Table 3.5.2, because models in this comparison setting have 9 observed variables, the fitting results in uniform random correlation matrices for both models are not very good. The mean values for Model 5A and 5B are very close, and Model 4B has slightly higher frequency of data sets fit better than 5A for most fit indices (except GFI). Histograms and CDF plots confirmed these results are consistent. Because neither of them can fit well in random uniform correlation matrices, results from known model analysis are very important for this comparison. In random population correlation matrices generated from Model 5A, use p-value from chi-square test as fit index, Model 5B fits well in 11.76% (1,176 out of 10,000) cases. In correlation matrices generated from Model 5B, Model 5A fits well in 2.89% (289 out of 10,000) cases. These results showed Model 5B is a more complex model even though it has same model degrees of freedom with Model 5A and smaller number of model implied vanishing tetrad.

In comparison 5, we have demonstrated that when compare the complexity of two models which are not nested in terms of vanishing tetrads, using the number of vanishing tetrads to indicate model complexity is not always correct.

3.6 Complexity Due to Constraint Type

In SEM models, different constraint types can be used between variables, for example, fixed-value constraints and equality constraints are often used. It is reasonable to assume that equality constraints will have different effects on model complexity compared to fixed-value constraint. Moreover, when a variable only has one constraint to other variables in the model, if we set the value of this constraint equals to zero, this variable will be independent from other variables in the model. It is expected that this special case will have the lowest complexity among those constraint types.

Our expectation about the relationships between model complexity and constraint types can be tested by fitting 3 models to the same data. Model 6A, 6B and 6C shown in Figure 3.6.1, Figure 3.6.2 and Figure 3.6.3 have same functional forms, same model degrees of freedom (*df* = 7) but different constraint types. In Model 6A, the effect between x_4 and x_5 is fixed to 0.5. In Model 6B, the constraints between x_2 and x_4, x_3 and x_4 have equal values. In Model 6C, the effect between x_4 and x_5 is fixed to 0.

Figure 3.6.1 Model 6A, effect between x_4 and x_5 is fixed to 0.5

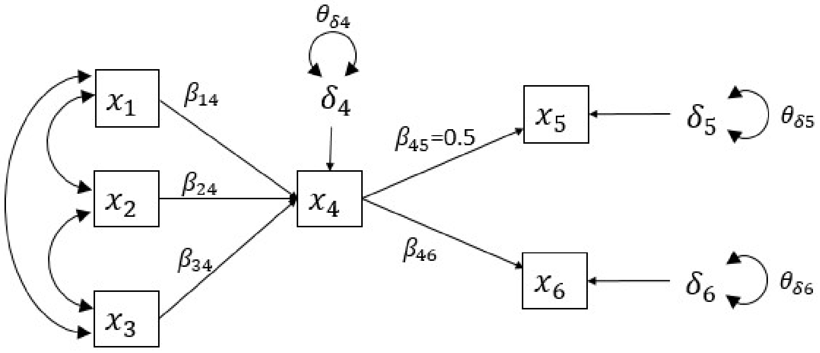


Figure 3.6.2 Model 6B, effects between x_4 and x_5, x_4 and x_6 are equal

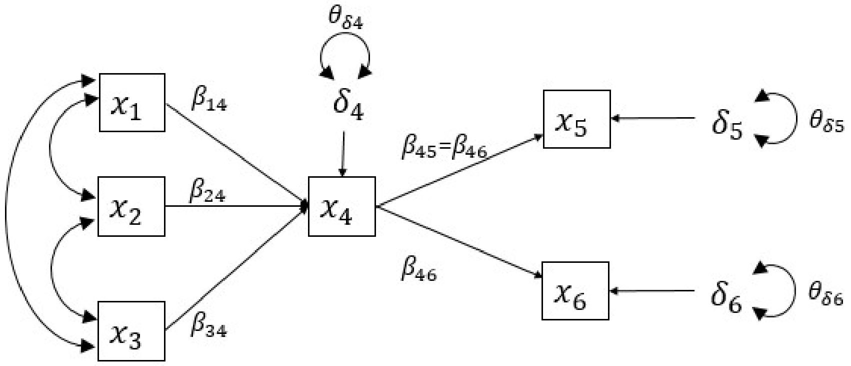
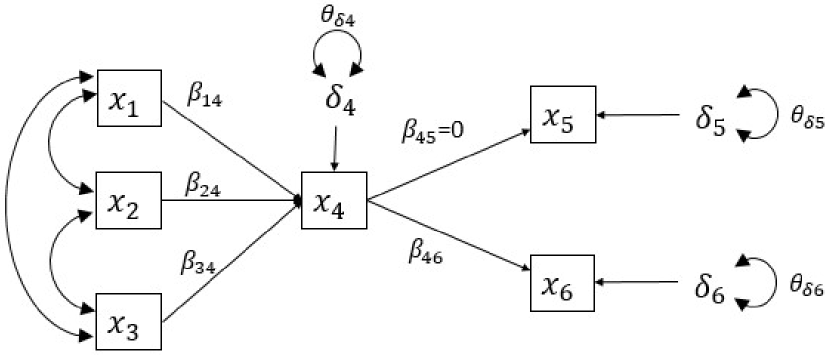


Figure 3.6.3 Model 6C, effect between x_4 and x_5 is fixed to 0



In uniform random correlation matrices analysis, all three models had 100% convergence rate. Fitting results for Model 6A and 6B in these matrices are listed in Table 3.6.1, and results for Model 6A and 6C are listed in Table 3.6.2. Histograms and cumulative frequency distribution (CDF) plots of fitting results are showed in Figure 3.6.4 and Figure 3.6.5.

Table 3.6.1 Fitting results for Model 6A and 6B in 10,000 uniform random correlation matrices.

	P-value of χ^2 test		**SRMR**		**RMSEA**		**GFI**	
Threshold Level	> 0.05		< 0.08		< 0.1		> 0.95	
	6A	**6B**	**6A**	**6B**	**6A**	**6B**	**6A**	**6B**
Mean	0.000	0.000	0.193	0.178	0.493	0.491	0.746	0.758
SD	0.000	0.000	0.003	0.003	0.024	0.024	0.005	0.004
N of fit well	15	14	147	284	15	14	23	20
N of fit better	809	1099	2766	7234	4031	5969	3171	6829
	CFI		**TLI**		**AIC**		**BIC**	
Threshold Level	> 0.95		> 0.95					
	6A	**6B**	**6A**	**6B**	**6A**	**6B**	**6A**	**6B**
Mean	0.544	0.549	0.023	0.034	1507.786	1506.710	1544.258	1543.182
SD	0.032	0.030	0.146	0.140	7839.187	7379.284	7839.187	7379.284
N of fit well	41	31	41	31				
N of fit better	4031	5969	4031	5969	4031	5969	4031	5969

Table 3.6.2 Fitting results for Model 6A and 6C in 10,000 uniform random correlation matrices.

	P-value of χ^2 test		**SRMR**		**RMSEA**		**GFI**	
Threshold Level	> 0.05		< 0.08		< 0.1		> 0.95	
	6A	**6C**	**6A**	**6C**	**6A**	**6C**	**6A**	**6C**
Mean	0.000	0.000	0.193	0.204	0.493	0.503	0.746	0.755
SD	0.000	0.000	0.003	0.003	0.024	0.023	0.005	0.004
N of fit well	15	7	147	141	15	7	23	14
N of fit better	1166	899	6350	3650	5404	4596	4147	5853
	CFI		**TLI**		**AIC**		**BIC**	
Threshold Level	> 0.95		> 0.95					
	6A	**6C**	**6A**	**6C**	**6A**	**6C**	**6A**	**6C**
Mean	0.544	0.528	0.023	-0.011	1507.786	1514.508	1544.258	1550.98
SD	0.032	0.031	0.146	0.140	7839.187	7252.617	7839.187	7252.617
N of fit well	41	21	41	21				
N of fit better	5404	4596	5404	4596	5404	4596	5404	4596

Figure 3.6.4. Histograms and CDF plots of fitting results for Model 6A (Red) and 6B (Blue) to 10,000 uniform random correlation matrices

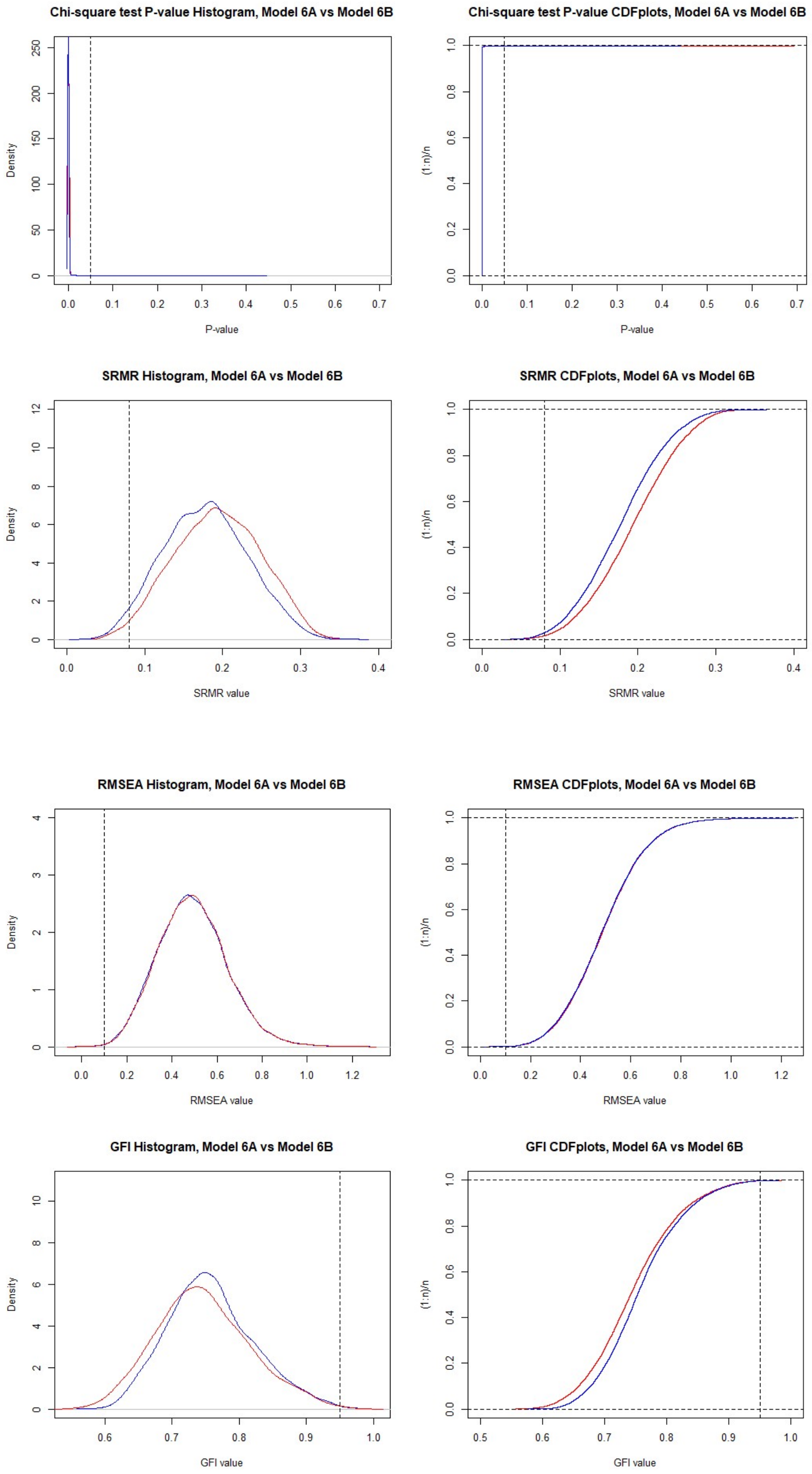


Figure 3.6.4. Histograms and CDF plots of fitting results for Model 6A (Red) and 6B (Blue) to 10,000 uniform random correlation matrices

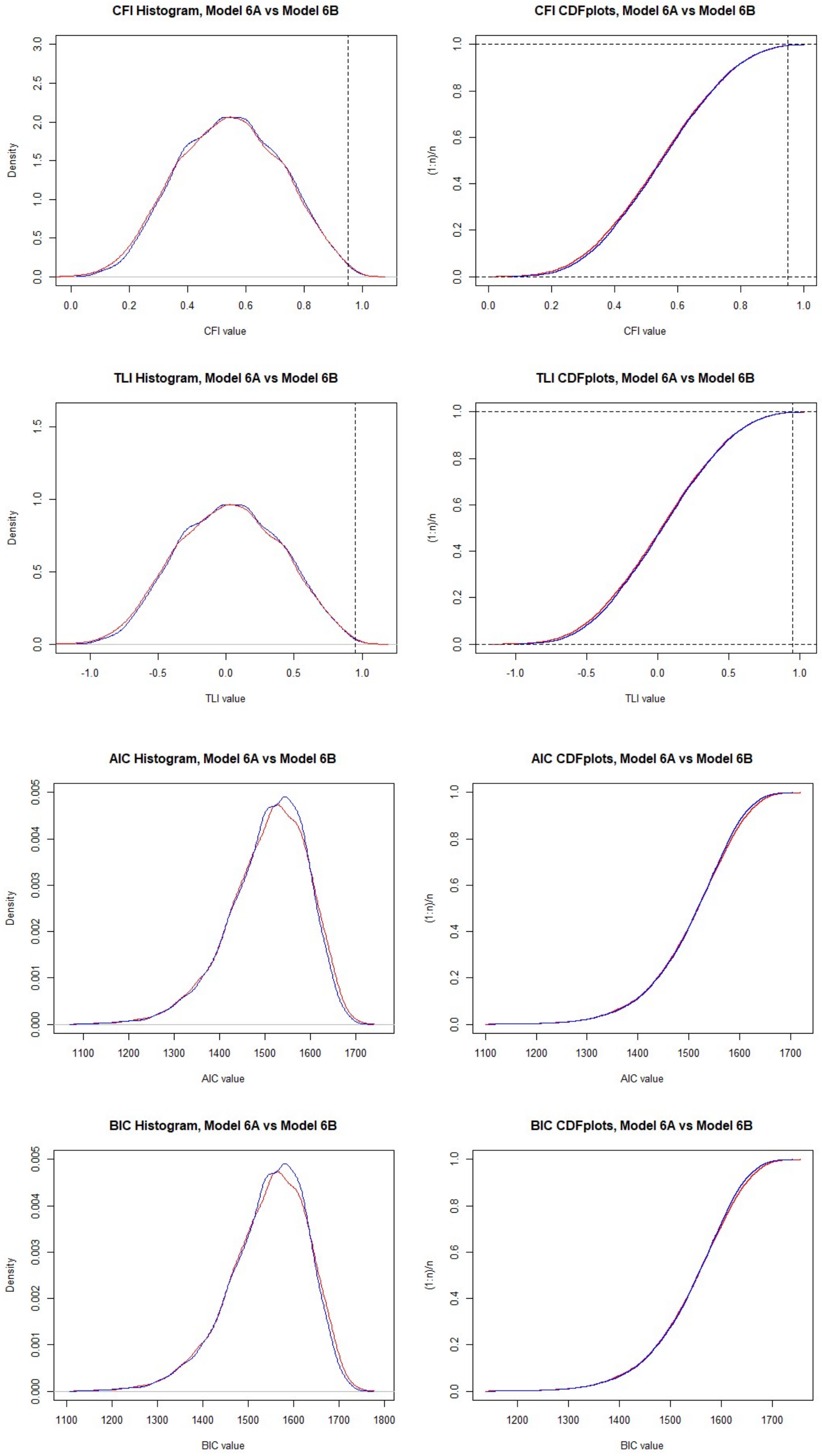


Figure 3.6.5. Histograms and CDF plots of fitting results for Model 6A (Red) and 6C (Blue) to 10,000 uniform random correlation matrices

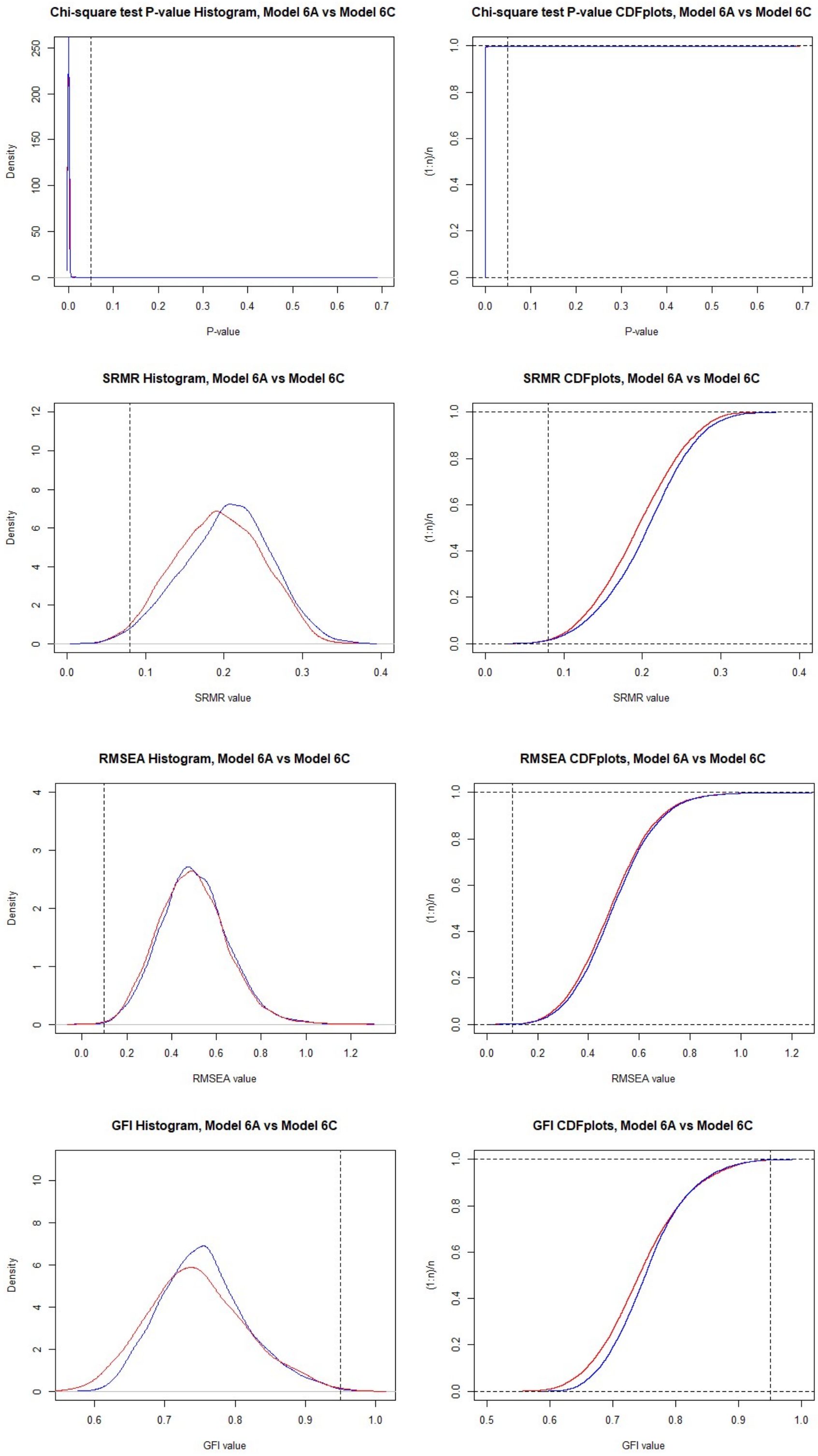
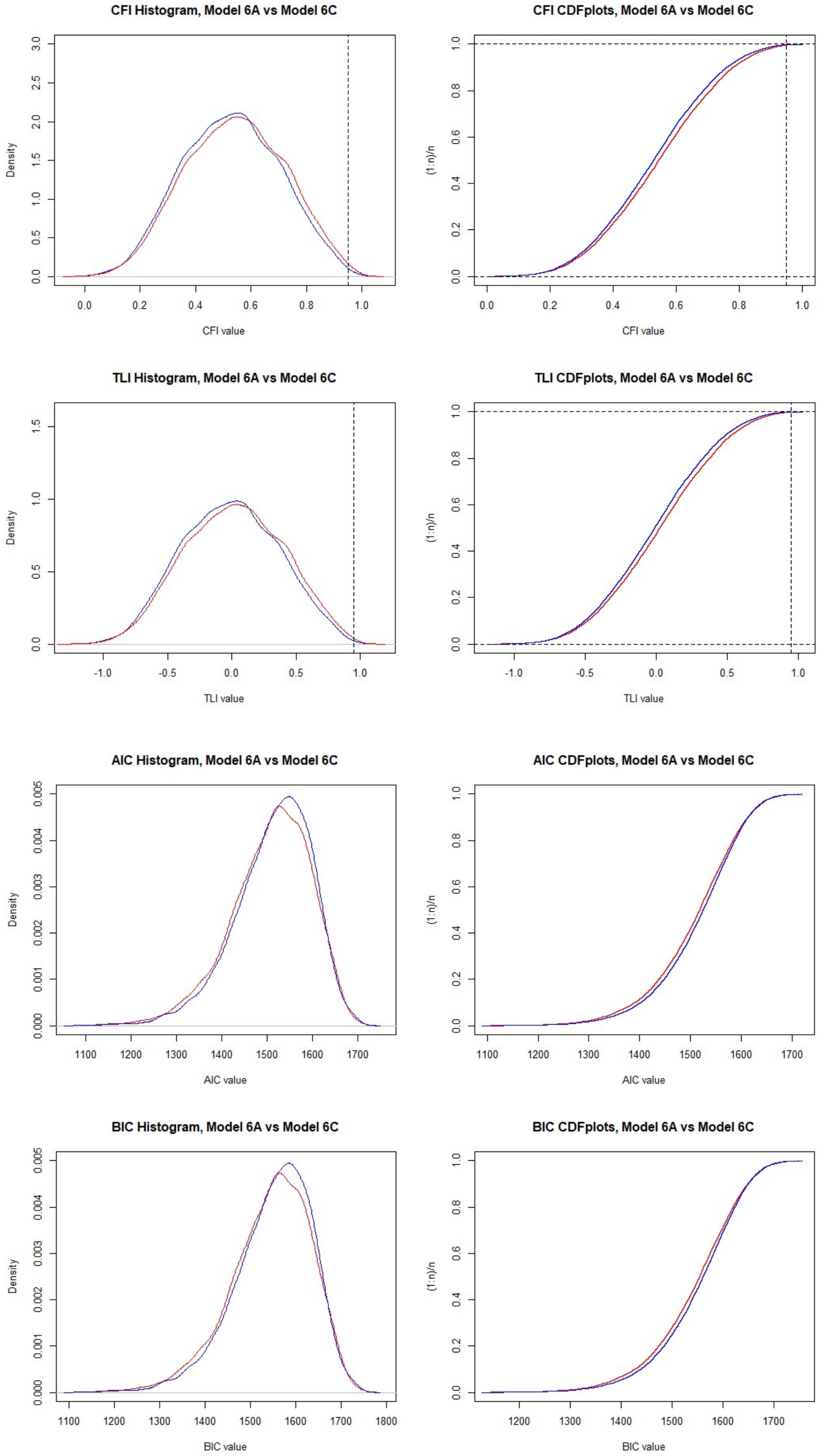


Figure 3.6.5. Histograms and CDF plots of fitting results for Model 6A (Red) and 6C (Blue) to 10,000 uniform random correlation matrices



From fitting results showed in Table 3.6.1, Model 6B has better mean values than Model 6A in almost all fit indices and it has more number of better fit than 6A in every fit indices, although 6B has lower frequency of data sets fit well in some fit indices, we can say Model 6B fits those uniform random correlation matrices better than Model 6A, and this difference is clearly showed in Figure 3.6.4.

Comparison results between Model 6A and 6C are shown in Table 3.6.2. In this table, we can see Model 6A has better mean values, a higher frequency of data sets fit well and a higher frequency of data sets fit better than Model 6C in almost every fit index (except GFI), which indicates Model 6A is more complex than Model 6C.

As expected, from the fitting results to uniform random correlation matrices, among these three models with different constraints, the model with equality constraints (Model 6B) is the most complex model, the model with fixed value constraint (Model 6A) is the second complex model and the model with a zero constraint (Model 6C) is the least complex model. Thus, we demonstrated that different constraint types can result in different model complexity.

In model complexity due to different constraint type situations, model implied vanishing tetrad numbers are not useful to indicate model complexity. Because in tetrad analysis, set constraints fixed values or equal values will not change model implied vanishing tetrad numbers in most times. For example, τ_{1235} showed below is one tetrad should be tested in Model 6A, 6B and 6C, and it is not equal to zero before we set special constraints to the model. We can see how different constraint types can affect its value.

$$\tau_{1235} = \sigma_{12}\sigma_{35} - \sigma_{13}\sigma_{25} = \phi_{12} * (\beta_{34} * \beta_{45} * \phi_{33} + \beta_{24} * \beta_{45} * \phi_{23} + \beta_{14} * \beta_{45} * \phi_{13})$$

$$-\phi_{13} * (\beta_{24} * \beta_{45} * \phi_{22} + \beta_{14} * \beta_{45} * \phi_{12} + \beta_{34} * \beta_{45} * \phi_{23}) \qquad (3.6.1)$$

In Model 6A, we set $\beta_{45} = 0.5$, so model implied τ_{1235} value will change to:

$$\tau_{1235} = \sigma_{12}\sigma_{35} - \sigma_{13}\sigma_{25} = \beta_{45} * \phi_{12} * (\beta_{34} * \phi_{33} + \beta_{24} * \phi_{23} + \beta_{14} * \phi_{13})$$

$$-\beta_{45} * \phi_{13} * (\beta_{24} * \phi_{22} + \beta_{14} * \phi_{12} + \beta_{34} * \phi_{23})$$

$$= 0.5 * \phi_{12} * (\beta_{34} * \phi_{33} + \beta_{24} * \phi_{23} + \beta_{14} * \phi_{13})$$

$$-0.5 * \phi_{13} * (\beta_{24} * \phi_{22} + \beta_{14} * \phi_{12} + \beta_{34} * \phi_{23}) \quad (3.6.2)$$

Thus, the model implied τ_{1235} value in Model 6A will not equal to zero although we set fixed value for β_{45}.

In Model 6B, we set $\beta_{45} = \beta_{46}$, so model implied τ_{1235} will remain the same since β_{46} is not included in the equation 3.6.1.

In Model 6C, we set $\beta_{45} = 0$, which is a strong assumption, means variable x_5 is not correlated to any other observed variables in the model. Under this assumption σ_{15}, σ_{25}, σ_{35}, and σ_{45} should all equal to 0, thus $\tau_{1235} = \sigma_{12}\sigma_{35} - \sigma_{13}\sigma_{25} = 0$, that means τ_{1235} should be a vanishing tetrad in Model 6C.

When no constraints are placed on the model, there are six model implied vanishing tetratds ($\tau_{1452}, \tau_{1462}, \tau_{1453}, \tau_{1456}, \tau_{1456}, \tau_{1645}$). When we set $\beta_{45} = 0.5$ in Model 6A and set $\beta_{45} = \beta_{46}$ in Model 6B, model implied vanishing tetrads remain the same. In Model 6C, we set $\beta_{45} = 0$ and it causes many model implied covariance values to equal 0, so there are 7 vanishing tetrads ($\tau_{1235}, \tau_{1523}, \tau_{1245}, \tau_{1524}, \tau_{1462}, \tau_{1256}, \tau_{1463}$) implied in this model. Moreover, the 6 vanishing tetrads implied by Model 6A and 6B are nested in those 7 vanishing tetrads implied by Model 6C.

In this comparison, vanishing tetrad numbers cannot recognize the difference in model complexity (Model 6A vs Model 6B) and the model with more vanishing tetrad numbers is not a more complex, even though models are nested in terms of vanishing tetrads (Model 6A or 6B vs Model 6C). So vanishing tetrad number cannot be used to indicate model complexity when model complexity is caused by different constraint types.

3.7 Summary

This chapter introduced a new two-step method of comparing model complexity in uniform random correlation matrices first then use the fitting results from known model implied population matrices to confirm the difference we found in the first step. With mean values, the frequency of data sets fit well and the frequency of data sets fit better using many commonly used SEM fit indices, we can get a comprehensive understanding about how well the compared models can fit random data. In this chapter, when comparing model complexity in fitting results from uniform random correlation matrices, the model that fits better in more fit indices is considered as a more complex model.

By comparing model complexity of several comparative SEM models, we demonstrated that: (1) Vanishing tetrad numbers can be used to recognize different functional forms in SEM models; and (2) Vanishing tetrad numbers can be used to indicate model complexity when two SEM models have same model degree of freedom but different functional form and models are nested in terms of vanishing tetrads. In this situation, nested tetrad analysis can be used to compare model fit. The model with more vanishing tetrads is a more complex model. (3) When two models are not nested in terms of vanishing tetrads, using vanishing tetrad number to indicate model complexity is not always correct, and researchers should not determine model complexity only by vanishing tetrad numbers (4) Vanishing tetrad number cannot be used to indicate model complexity when model complexity is caused by different constraint types.

Chapter 4 Application

4.1 Introduction

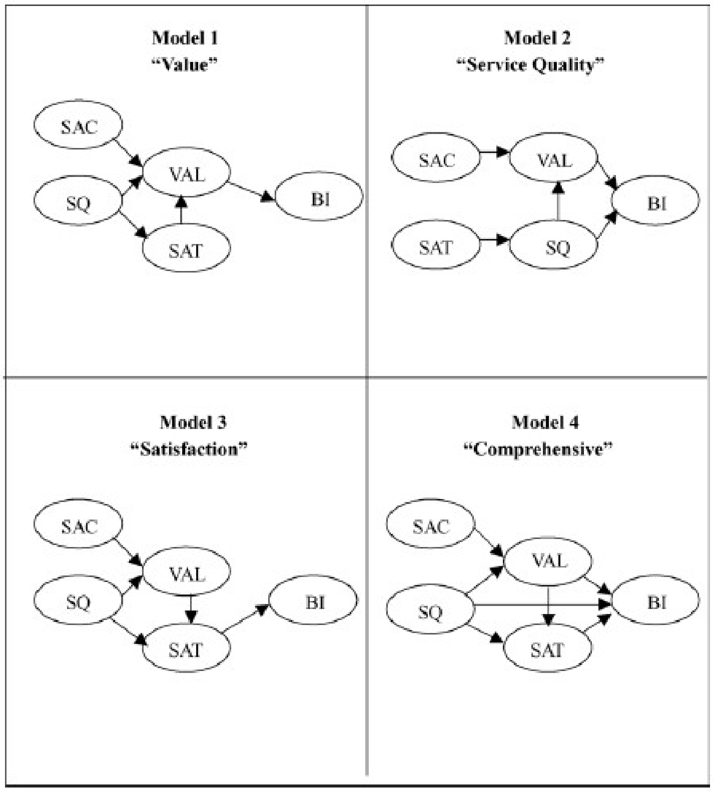
In Chapter 3, several comparisons between different SEM models were used to demonstrate the possibility of using model implied vanishing tetrad number to indicate model complexity when models have same model degrees of freedom. Results lead to the conclusion that when models are nested in terms of vanishing tetrads, model with more vanishing tetrads should be the more complex model. When models are not nested in terms of vanishing tetrads, vanishing tetrad number can be used in already compared functional forms to indicate model complexity.

In substantive SEM studies, different functional forms are often used and compared, and improperly accounting for model complexity may lead to wrong decisions about the best model of the study. Therefore, an example was found in published research to illustrate how model complexity analysis and vanishing tetrad numbers can work to improve model selection.

4.2 A Substantive Model Selection Example

In Brady (2005), four service evaluation models showed in Figure 4.2.1 were identified that are commonly offered to depict the relationships amongst the primary service evaluation constructs of sacrifice (SAC), service quality (SQ), service value (VAL), satisfaction (SAT), and behavioral intentions (BI). The author tried to determine the best fitting model by testing these four models using samples of service consumers in Australia, Hong Kong, Morocco, the Netherlands, and the United States, as well as across varied temporal and service settings.

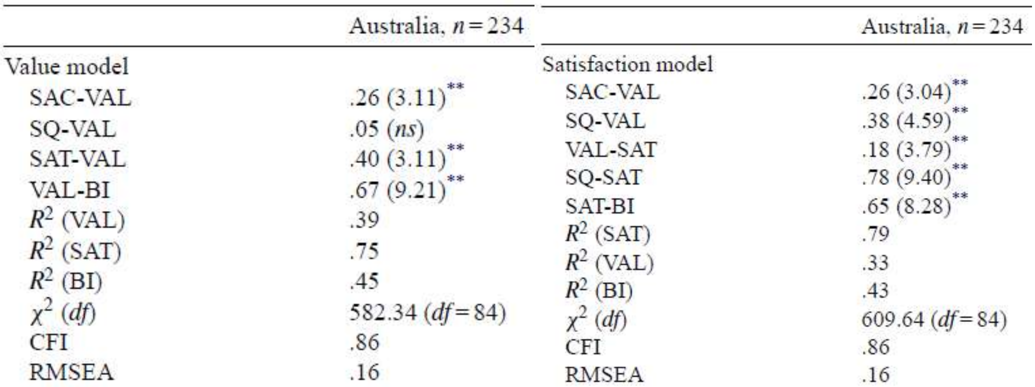
Figure 4.2.1 Four service evaluation models from Brady (2005)



Among four service evaluation models from Brady (2005), the “Value” model and “Satisfaction” model both have 84 model degrees of freedom (Including 4 degrees of freedom from structural model and 80 degrees of freedom from indicators which are omitted in Figure 4.2.1) and they are not nested in traditional ways. However, in section 3.5, we already demonstrated the functional form in the “Value” model has more vanishing tetrads than the functional form in the “Satisfaction” model, thus the “Value” model should be considered a more complex than the “Satisfaction” model.

Results for the “Value” and “Satisfaction” models in Australian sample are shown in Table 4.2.1. Two models had the same CFI and RMSEA values and the “Value” model has a lower chi-square value. Thus, in Brady (2005), the “Value” model is considered as a better fit model than “Satisfaction” model in Australia sample.

Table 4.2.1 Model fitting results for Value model and Satisfaction model in Australia sample, from Brady (2005)



	Australia, $n = 234$
Value model	
SAC-VAL	.26 (3.11)**
SQ-VAL	.05 (*ns*)
SAT-VAL	.40 (3.11)**
VAL-BI	.67 (9.21)**
R^2 (VAL)	.39
R^2 (SAT)	.75
R^2 (BI)	.45
χ^2 (*df*)	582.34 ($df = 84$)
CFI	.86
RMSEA	.16

	Australia, $n = 234$
Satisfaction model	
SAC-VAL	.26 (3.04)**
SQ-VAL	.38 (4.59)**
VAL-SAT	.18 (3.79)**
SQ-SAT	.78 (9.40)**
SAT-BI	.65 (8.28)**
R^2 (SAT)	.79
R^2 (VAL)	.33
R^2 (BI)	.43
χ^2 (*df*)	609.64 ($df = 84$)
CFI	.86
RMSEA	.16

Using model complexity analysis, we can show that the “Value” model can fit random data better than the “Satisfaction” model if we use CFI and RMSEA as fit indices (see Table 3.5.1).

Therefore, when two models have the same CFI and RMSEA results, “Satisfaction” model should be preferred, which means we made a different conclusion as compared to Brady (2005).

In this case, since two models have very similar fitting results, the model selection result is highly depending on model complexity analysis result. Model implied tetrad number and model complexity method from our study can be very helpful to determine which model is better.

Chapter 5 Conclusions and Discussion

5.1 Summary

This dissertation has introduced and further demonstrated the inadequacy of traditional fit indices in structural equation modeling found by Preacher 2006, which didn't always account for model complexity correctly. Chapter 1 introduced the basics of structural equation modeling, explained how traditional fit indices measure model fit, and why they are not adequate. Also, confirmatory tetrad analysis, a technique to estimate SEM model fit in another way was introduced, and the model implied vanishing tetrad number was treated as a potential index of model complexity. Chapter 2 introduced different methods to determine SEM model complexity using a large number of simulated data and three different ways to generate random samples. I then explained how to investigate the relationship between model complexity and tetrad number. Chapter 2 also introduced a new R package to perform tetrad analysis and explained why a two-step method is preferred in this study to compare SEM model complexity. Chapter 3 used the method in Chapter 2 to compare model complexity between SEM models which have same degrees of freedom, then demonstrated that tetrad number can be used in some scenarios to indicate model complexity and should be tested in more cases. Chapter 4 applied complexity comparison method in Chapter 2 in real models chosen from a published paper and analyzed model implied vanishing tetrads to demonstrate how complexity analysis and tetrad number can improve model selection in an SEM study.

5.2 Primary Findings

First of all, from all model complexity comparison results between SEM models with the same degrees of freedom but different functional forms or different constraint types, the inadequacy of traditional SEM model fit indices was demonstrated. Comparison results showed that only using model degrees of freedom to measure SEM model complexity is not always correct, and it is necessary to consider the effect of other factors such as functional form and constraint type in model selection. Secondly, a two-step method to evaluate/compare SEM model complexity was presented in Chapter 2 and applied to all comparison examples in this study, and all these comparison results showed this method is useful in some situations. Thirdly, an R package to run confirmatory tetrad analysis was developed in Chapter 2. It can be used to get model implied vanishing tetrad numbers, test if models are nested in terms of tetrads and run test of model fit in tetrad analysis. Fourthly, the possibility of using the model implied vanishing tetrad number as another index of SEM model complexity was examined in Chapter 2. Comparisons 1 to 5 demonstrated that different functional forms can affect model complexity and also vanishing tetrad number, for models that have the same degrees of freedom and are nested in terms of tetrads, models with more vanishing tetrads should be more complex. For models not nested in terms of tetrads, this rule is not always true. Comparison 6 demonstrated that SEM models with different constraint types may have different model complexity but neither degrees of freedom nor tetrad number can be used to detect this difference.

5.3 Implications and Recommendations for SEM Model Selection

On the purpose of selecting the best model, researchers should do their best to evaluate the goodness of fit and model complexity. In this study, we demonstrated that a model that performs better in one fit index may perform worse in another. Thus, the more evidence researchers can find, the more confident they will be, about their model selection results. Compared to goodness of fit, model complexity is hard to quantify and always evaluated crudely in traditional SEM fit indices (e.g. only evaluated by number of free parameters). Therefore, improvement of the evaluation of SEM model complexity is necessary.

Confirmatory tetrad analysis is recommended in model selection because it provides another way to measure the goodness of model fit. Models not nested according to the traditional LR test may be nested in terms of tetrads, and different tetrad number may suggest different model complexity. R package created in this study is easy to access and not hard to use, to run confirmatory tetrad analysis. Therefore, tetrad analysis can be treated as a complement of traditional fit indices to do model selection, because when models have different tetrad numbers or different tetrad settings, they may have different model complexity. When models are nested in terms of tetrads, nested tetrad test can be used to do model selection and tetrad number can be used as another index of model complexity. However, the correctness of using tetrad number as another index of model complexity has not been fully verified. Although all comparisons between models nested in terms of tetrads in this study showed models with more vanishing tetrads are more complex, the relationship between SEM model complexity and number of vanishing tetrads is not fully clear. For example, nested tetrad test results for Model 3A and 3B in section 3.4 showed 3A fitted better while all other fit indices showed Model 3B fitted better, thus nested tetrad test results were not treated as a crucial standard to determine the goodness of model fit in this case. Therefore, neither tetrad analysis nor traditional measures can fully account for SEM model complexity in every situation.

The two-step method presented in this study is highly recommended to quantify and compare the model complexity. Although it needs several hours to conduct, it can provide more complete and more defensible result than only use CTA or traditional fit indices to evaluate model complexity. Chapter 5 showed a successful example of how to use this two-step method to improve model selection in research. Moreover, when considering SEM model selection, determination of goodness of model fit and model complexity are not only factors researchers should keep in mind. Other factors such as theoretical meaning of the model must also be considered because SEM models are used to explain the relationships between variables. Therefore, models without theoretically meaningful explanations should not be selected even if they can fit the data perfectly. When models have close goodness of fit and model complexity, think about which model is more theoretical reasonable for the study may be a good way to determine which model should be selected.

5.4 Limitations

Due to computational power, the comparison results using uniform random correlation matrices may not be very accurate because some models have close fitting results and factors like random seed used in data generation may affect fitting results. A larger sample and using different random seeds would result in more accurate results. Therefore, comparison results from known-model population correlation matrices must be used in every case to support or against the comparison results from uniform random correlation matrices.

In known-model data generation, all linear coefficients were set as uniform distributed variables in range (0.05, 0.95) instead of range (0.00, 1.00) because linear coefficients closer to 0 may be considered as non-significant parameters in SEM model analysis and parameter values closer to 1 will make the variance of its random error term closer to 0. Different range of linear coefficient values may affect fitting results in known-model analysis.

Moreover, in order to get a better understanding about the relationship between tetrad number and SEM model complexity, more comparisons of SEM models would be useful, especially when models have same degrees of freedom and nested in terms of tetrads.

5.5 Directions for Future Research

In the future research of SEM model comparison, using computational power or developing a new index of model fit are two possible ways to determine model complexity. Using computational power to determine SEM model complexity by fitting models to large random samples is a safe but slow way to improve model selection. Future research should improve this method by using larger samples and increasing efficiency. Increased computing power, faster data generation methods and faster model fit processes should be very helpful.

Another possible way to improve SEM model complexity measurement is trying to find or create a new index of model complexity. For example, tetrad number has been demonstrated as a possible index of complexity, especially when models are nested in terms of tetrads. Future research should test more models nested in terms of tetrads to demonstrate this possibility and if it is always true, quantify the relationship between model complexity and tetrad number, then find a way to combine tetrad number and traditional fit indices to create better indices of SEM model fit.

List of References

Akaike, H. (1974). A new look at the statistical model identification. IEEE transactions on automatic control, 19(6), 716-723.

Bentler, P. M. (1990). Comparative fit indexes in structural models. Psychological bulletin, 107(2), 238.

Bentler, P. M., & Bonett, D. G. (1980). Significance tests and goodness of fit in the analysis of covariance structures. Psychological bulletin, 88(3), 588.

Bollen, K. A. (1989). A new incremental fit index for general structural equation models. Sociological Methods & Research, 17(3), 303-316.

Bollen, K. A. (1990). Outlier screening and a distribution-free test for vanishing tetrads. Sociological Methods & Research, 19(1), 80-92.

Bollen, K. A. (2014). Structural equations with latent variables. John Wiley & Sons.

Bollen, K. A., & Ting, K. F. (1993). Confirmatory tetrad analysis. Sociological methodology, 147-175.

Bollen, K. A., & Ting, K. F. (1998). Bootstrapping a test statistic for vanishing tetrads. Sociological Methods & Research, 27(1), 77-102.

Bollen, K. A., & Ting, K. F. (2000). A tetrad test for causal indicators. Psychological methods, 5(1), 3.

Brady, M. K., Knight, G. A., Cronin Jr, J. J., Tomas, G., Hult, M., & Keillor, B. D. (2005). Removing the contextual lens: A multinational, multi-setting comparison of service evaluation models. Journal of Retailing, 81(3), 215-230.

Collyer, C. E. (1985). Comparing strong and weak models by fitting them to computer-generated data. Attention, Perception, & Psychophysics, 38(5), 476-481.

Cutting, J. E. (2000). Accuracy, scope, and flexibility of models. Journal of Mathematical Psychology, 44(1), 3-19.

Glymour, C., Scheines, R., Spirtes, P., & Kelly, K. (1987). Discovering Causal Structure: Artificial Intelligence. Philosophy of science, and Statistical Modeling, 205-212.

Hipp, J. R., & Bollen, K. A. (2003). Model fit in structural equation models with censored, ordinal, and dichotomous variables: Testing vanishing tetrads. Sociological Methodology, 33(1), 267-305.

Homburg, C. (1991). Cross-validation and information criteria in causal modeling. Journal of Marketing Research, 137-144.

Hooper, D., Coughlan, J., & Mullen, M. (2008). Structural equation modelling: Guidelines for determining model fit. Articles, 2.

Hox, J. J., & Bechger, T. M. (1998). An introduction to structural equation modelling. Family Science Review, 11(354-373).

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.

Joe, H. (2006). Generating random correlation matrices based on partial correlations. Journal of Multivariate Analysis, 97(10), 2177-2189.

Jöreskog, K. G. (1967). A general approach to confirmatory maximum likelihood factor analysis. ETS Research Bulletin Series, 1967(2), 183-202.

Jöreskog, K. G. (1970). A general method for estimating a linear structural equation system. ETS Research Bulletin Series, 1970(2), i-41.

Jöreskog, K. G. (1993). Testing structural equation models. Sage focus editions, 154, 294-294.

Kenny, D. A., Kaniskan, B., & McCoach, D. B. (2014). The performance of RMSEA in models with small degrees of freedom. Sociological Methods & Research, 0049124114543236.

Kline, R. B. (2015). Principles and practice of structural equation modeling. Guilford publications.

Lei, P. W., & Wu, Q. (2007). Introduction to structural equation modeling: Issues and practical considerations. Educational Measurement: issues and practice, 26(3), 33-43.

Moss, S. (2009). Fit indices for structural equation modeling. Website: http://www. psych-it. com. au/Psychlopedia/article. asp.

Newsom, J. T. (2012). Some clarifications and recommendations on fit indices. USP, 655, 123-133.

Paxton, P., Curran, P. J., Bollen, K. A., Kirby, J., & Chen, F. (2001). Monte Carlo experiments: Design and implementation. Structural Equation Modeling, 8(2), 287-312.

Preacher, K. J. (2003). The role of model complexity in the evaluation of structural equation models (Doctoral dissertation, The Ohio State University).

Preacher, K. J. (2006). Quantifying parsimony in structural equation modeling. Multivariate Behavioral Research, 41(3), 227-259.

Preacher, K. J., & Merkle, E. C. (2012). The problem of model selection uncertainty in structural equation modeling. Psychological methods, 17(1), 1.

Roberts, S., & Pashler, H. (2000). How persuasive is a good fit? A comment on theory testing. Psychological review, 107(2), 358.

Rosseel, Y. (2012). lavaan: An R package for structural equation modeling. Journal of Statistical Software, 48(2), 1-36.

Schwarz, G. (1978). Estimating the dimension of a model. The annals of statistics, 6(2), 461-464.

Spearman, C. (1904). “General Intelligence," objectively determined and measured. The American Journal of Psychology, 15(2), 201-292.

Stanley, J. C., & Wang, M. D. (1969). Restrictions on the possible values of r12 given r13 and r23. Educational and Psychological Measurement.

Wright, S. (1934). The method of path coefficients. The annals of mathematical statistics, 5(3), 161-215.

APPENDIX A

R CODES FOR CONFIRMATORY TETRAD ANALYSIS

The R codes in Appendices A, B, and C were written and compiled using R 3.3.4.

```
install.packages("lavaan")
############################################################################################
############################### Tetrad Function ###########################################
############################################################################################
TetradAnalysisNoRandom<-function(Samplecov,FactorModel,Size){
### "sweep_all" function is necessary to find Nonredundant vanishing tetrads from empirical vanishing
tetrads ###
sweep<-function(A,k) {
        B<-matrix(0,nrow=nrow(A),ncol=ncol(A))
        if (abs(A[k,k])<2.2*10^(-10)){
                B=A
                for (i in 1:nrow(B)){
                        B[i,k]=0
                        B[k,i]=0
                }
        }
        else if (A[k,k] !=0) {
                for (i in 1:nrow(A)) {
                        for (j in 1: ncol(A)) {
                        if (i== k & j==k){
                                B[i,j]=1/A[k,k]
                                }
                        else if (i == k & j != k) {
                                B[i,j] = A[k,j]/A[k,k]
                                }
                        else if (i != k & j == k) {
                                B[i,j] = -A[i,k]/A[k,k]
                                }
                        else if (i != k & j != k) {
                                B[i,j] = A[i,j] - ( A[i,k]*A[k,j] / A[k,k])
                                }
                        }
                }
        }
        return(B)
}

sweep_all<-function(A) {
        B<-matrix(0,nrow=nrow(A),ncol=ncol(A))
        for (i in 1: nrow(A)) {
                B=sweep(A,i)
                A=B
        }
        return(B)
}
#####################################################################################################
### Tetrad Function Part 1: Use Empirical Method to find model implied tetrad #################
#####################################################################################################
### Information needed for Part 1 ###
### 1. Sample covariance matrix "Samplecov". ###
### 2. SEM model need to be tested "FactorModel". ###
### 3. Sample size "Size". ###
### Use function "ExampleTetrad" to get all model implied vanishing tetrads using empirical method.
###
EmpiricalMethod<-function(FactorModel,Samplecov){
### Package "lavaan" is used to get model implied covariance matrix "Modelcov", the synatax used to
sepecify model is the same as syntax in "lavaan". ###
        library(lavaan)
        FactorOut<-cfa(FactorModel, sample.cov=Samplecov, sample.nobs =Size)
        Modelcov<-data.frame(lavTech(FactorOut, "cov.ov"))*Size/(Size-1)
### List all tetrads in matrix "Tetrad", when there are N observed variables in inputted model. ###
        N<-ncol(Modelcov)
        x<-combn(1:N,4)
        TetradNum<-choose(N, 4)
        x1<-t(x)
        x2<-cbind(x[1,],x[4,],x[2,],x[3,])
        x3<-cbind(x[1,],x[3,],x[4,],x[2,])
        y<-rbind(x1,x2,x3)
        rownames(y)<-rep(1:TetradNum,3)
        Y<-y[order(as.numeric(rownames(y))),]
        Tetrad<-Y[,1]*1000+Y[,2]*100+Y[,3]*10+Y[,4]
### Calculate all tetrad values based on model implied covariance matrix "Modelcov". ###
### Save all tetrad values in a vector named "T", if the absolute tetrad value is less than 0.001, we
assume it is a vanishing tetrad. ###
        T<-matrix(1,nc=1,nr=nrow(Y))
        for(i in 1:nrow(Y)){
        T[i,1]<-round(Modelcov[Y[i,2],Y[i,1]]*Modelcov[Y[i,4],Y[i,3]]
                  -Modelcov[Y[i,3],Y[i,1]]*Modelcov[Y[i,4],Y[i,2]],2)
        }
        colnames(T) <- c("ImpliedValue")
        implied<-matrix(,nrow=nrow(T))
        colnames(implied) <- c("Implied")
```

```
          for(i in 1:nrow(T)){
          if(abs(T[i,1])>0.001) {implied[i,1]<-0} else{implied[i,1]<-1}
          }
          Tvalue<-matrix(1,nc=4,nr=nrow(Y))
          for (i in 1:nrow(Y)){
          Tvalue[i,1]<-min(Y[i,1],Y[i,2])*10+max(Y[i,1],Y[i,2])
          Tvalue[i,2]<-min(Y[i,3],Y[i,4])*10+max(Y[i,3],Y[i,4])
          Tvalue[i,3]<-min(Y[i,1],Y[i,3])*10+max(Y[i,1],Y[i,3])
          Tvalue[i,4]<-min(Y[i,2],Y[i,4])*10+max(Y[i,2],Y[i,4])
          }
          colnames(Tvalue) <- c("1","2","3","4")
          EmpiricalMethod_list<-list(EmpiricalMethod=cbind(Tetrad,Tvalue,implied,T),
Modelcov=Modelcov)
          return(EmpiricalMethod_list)
}
##################################################################################
########## Tetrad Function Part 2: Find Nonredundant vanishing tetrads ######################
##################################################################################
### Information needed for Part 2 ###
### 1. Model implied vanishing tetrads "ModelTetrad". ###
ExampleTetrad<-EmpiricalMethod(FactorModel,Samplecov)                                   # Use
"EmpiricalMethod" function. #
ModelTetrad<-ExampleTetrad$EmpiricalMethod[ExampleTetrad$EmpiricalMethod[,6]==1,]        #
Implied=1 means tetrad vanished in model. #
EmpiricalTetrads<-ModelTetrad[,1,drop=FALSE]                                # Model implied tetrad
names. #
### 2. All covariances in the model. ###
COV<-ModelTetrad[,2:5,drop=FALSE]
### 3. Model implied covariance matrix "Modelcov". ###
Modelcov<-ExampleTetrad$Modelcov
##################################################################################
### Use function "NRVT_Fun" to identify model implied nonredundant vanishing tetrads. ###
NRVT_Fun<-function(COV,ModelTetrad,Modelcov) {
          if (nrow(COV)==1){                    # If there is only one model implied vanishing tetrad,
identify is unnecessary. #
                    NRVT<-ModelTetrad[1,1]
                    NRVT_Num<-1
          }

          else if (nrow(COV)>1) {          # If there is more than one model implied vanishing tetrad,
identify is necessary. #
                    uc<-unique(as.vector(COV))       # Find all unique covariances among vanishing tetrads. #
                    SigmaSS<-matrix(1,nc=length(uc),nr=length(uc))
                    for (i in 1:length(uc)){
```

```
            for (j in 1:length(uc)) {
            e<-uc[i]%/%10
            f<-uc[i]%%10
            g<-uc[j]%/%10
            h<-uc[j]%%10
            efgh<-Modelcov[e,g]*Modelcov[f,h]+Modelcov[e,h]*Modelcov[f,g]
            SigmaSS[i,j]<-efgh
            }
        }
### Get model implied SigmaTT based on model implied covariance matrix "Modelcov". ###
        dtaudsigmaCOV<-matrix(0,ncol=length(uc),nrow=nrow(COV))
        for (i in 1:nrow(dtaudsigmaCOV)){
            for(j in 1:ncol(dtaudsigmaCOV)){
            if (COV[i,1]==uc[j])
{dtaudsigmaCOV[i,j]=Modelcov[COV[i,2]%/%10,COV[i,2]%%10]}
            if (COV[i,2]==uc[j])
{dtaudsigmaCOV[i,j]=Modelcov[COV[i,1]%/%10,COV[i,1]%%10]}
            if (COV[i,3]==uc[j]) {dtaudsigmaCOV[i,j]=-
Modelcov[COV[i,4]%/%10,COV[i,4]%%10]}
            if (COV[i,4]==uc[j]) {dtaudsigmaCOV[i,j]=-
Modelcov[COV[i,3]%/%10,COV[i,3]%%10]}
            }
        }
### use sweep operator to identify set of nonredundant vanishing tetrads. ###
        SigmaTT<-dtaudsigmaCOV %*% SigmaSS %*% t(dtaudsigmaCOV)
        SweepResults<-sweep_all(SigmaTT)
        rowsum<-rowSums(SweepResults)
        nrvt <- matrix(0, ncol=1, nrow=nrow(COV))
        NRVT_Num<-0
        for (i in 1: nrow(COV)) {
            if ( rowsum[i] != 0 ) {                   # If rowsum==0, that tetrad is
redundant.#
            nrvt[i,] = ModelTetrad[i,1]
            NRVT_Num<-NRVT_Num+1
                }
            }
    NRVT<-nrvt[nrvt[,1]!=0,]

    }
    NRVT_list<-list(NRVT=NRVT, NRVT_Num=NRVT_Num, uc=uc,Modelcov=Modelcov)
    return(NRVT_list)
}

NRVT_Results<-NRVT_Fun(COV,ModelTetrad,Modelcov)
```

```
NRVT_Num<-NRVT_Results$NRVT_Num
NRVT<-NRVT_Results$NRVT
UniqueCov<-NRVT_Results$uc

##################################################################################
########## Tetrad Function Part 3: Individual Test and Multivariate Test  ######################
##################################################################################
### Information needed for Part 3 ###
### 1. Model implied nonredundant vanishing tetrads "NRVT". ###
NRVT_Results<-NRVT_Fun(COV,ModelTetrad,Modelcov)            # Use "NRVT_Fun" function. #
NRVT<-NRVT_Results$NRVT
### 2. Model implied nonredundant vanishing tetrad number "NRVT_Num" ###
NRVT_Num<-NRVT_Results$NRVT_Num
### 3. All unique covariances in vanishing tetrads "uc". ###
uc<-NRVT_Results$uc

##################################################################################
######### Individual test for every nonredundant tetrad ###################################
Tetrad<-matrix(1,nc=1,nr=NRVT_Num)
AVAR<-matrix(1,nc=1,nr=NRVT_Num)
Teststat<-matrix(1,nc=1,nr=NRVT_Num)
Pvalue<-matrix(1,nc=1,nr=NRVT_Num)
Fun<-t(t(NRVT))                                # Get all nonredundant tetrads. #
NRVT_COV<-matrix(0, nrow=NRVT_Num,ncol=4)
for (k in 1:NRVT_Num){
                e<-Fun[k,1]%/%1000
                f<-Fun[k,1]%/%100%%10
                g<-Fun[k,1]%/%10%%10
                h<-Fun[k,1]%%10
        sigma<-
c(min(e,f)*10+max(e,f),min(g,h)*10+max(g,h),min(e,g)*10+max(e,g),min(f,h)*10+max(f,h))
        NRVT_COV[k,1]<-sigma[1]
        NRVT_COV[k,2]<-sigma[2]
        NRVT_COV[k,3]<-sigma[3]
        NRVT_COV[k,4]<-sigma[4]
        SigmaSS<-matrix(1,nc=length(sigma),nr=length(sigma))  # Get SigmaSS based on Sample
covariance matrix "Samplecov". #
                for (i in 1:length(sigma)){
                        for (j in 1:length(sigma)) {
                        ee<-sigma[i]%/%10
                        ff<-sigma[i]%%10
                        gg<-sigma[j]%/%10
                        hh<-sigma[j]%%10
                        efgh<-Samplecov[ee,gg]*Samplecov[ff,hh]+Samplecov[ee,hh]*Samplecov[ff,gg]
                SigmaSS[i,j]<-efgh
                        }
                }
dt<-matrix(c(Samplecov[g,h],Samplecov[e,f],-Samplecov[f,h],-Samplecov[e,g]),ncol=4,nrow=1)
VAR<-(1/Size)* dt %*% SigmaSS %*% t(dt)
T<-Samplecov[e,f]*Samplecov[g,h]-Samplecov[e,g]*Samplecov[f,h]                    # Calculate T values for
individual tests. #
stat<-T^2/VAR
Chipvalue<-1-pchisq(stat,df=1)                                          # Calculate P-values for
individual tests. #
        Tetrad[k,1]<-round(T,3)
        AVAR[k,1]<-round(VAR,3)
        Teststat[k,1]<-round(stat,3)
        Pvalue[k,1]<-round(Chipvalue,3)
}
Result<-cbind(Fun,Tetrad,AVAR,Teststat,Pvalue)
colnames(Result) <- c("Model Implied Tetrad","t","AVAR","TestStatistic","P-value")         # All results for
individual tests. #

###############################################################################
######### Multivariate test for overall model fit ###################################
### Get SigmaSS based on sample covariance matrix "Samplecov". ###
        SigmaSS<-matrix(1,nc=length(uc),nr=length(uc))
                for (i in 1:length(uc)){
                        for (j in 1:length(uc)) {
                        e<-uc[i]%/%10
                        f<-uc[i]%%10
                        g<-uc[j]%/%10
                        h<-uc[j]%%10
                        efgh<-Samplecov[e,g]*Samplecov[f,h]+Samplecov[e,h]*Samplecov[f,g]
                        SigmaSS[i,j]<-efgh
                                }
                        }

### Get SigmaTT, use dtau/dsigma and based on sample covariance matrix "Samplecov". ###
        dtaudsigmaCOV<-matrix(0,ncol=length(uc),nrow=NRVT_Num)
                for (i in 1:nrow(dtaudsigmaCOV)){
                        for(j in 1:ncol(dtaudsigmaCOV)){
                        if (NRVT_COV[i,1]==uc[j])
{dtaudsigmaCOV[i,j]=Samplecov[NRVT_COV[i,2]%/%10,NRVT_COV[i,2]%%10]}
                        if (NRVT_COV[i,2]==uc[j])
{dtaudsigmaCOV[i,j]=Samplecov[NRVT_COV[i,1]%/%10,NRVT_COV[i,1]%%10]}
                        if (NRVT_COV[i,3]==uc[j]) {dtaudsigmaCOV[i,j]=-
Samplecov[NRVT_COV[i,4]%/%10,NRVT_COV[i,4]%%10]}
```

```
                              if (NRVT_COV[i,4]==uc[j]) {dtaudsigmaCOV[i,j]=-
Samplecov[NRVT_COV[i,3]%/%10,NRVT_COV[i,3]%%10]}
                              }
                    }
SigmaTT<-dtaudsigmaCOV %*% SigmaSS %*% t(dtaudsigmaCOV)

t<-t(t(Result[,2]))                                       # Get T values for all individual tests, use function
"Result". #
T<-Size*t(t) %*% solve(SigmaTT) %*% t                     # T value for multivariate test. #
MultiPvalue<-1-pchisq(T,df=NRVT_Num)                      # P-value for multivariate test. #
colnames(T) <- c("TestStatistic For Multivariate Test")
colnames(MultiPvalue) <- c("P-value For Multivariate Test")

##################################################################################
########## Tetrad Function Part 4: Output all analysis results #################################
##################################################################################

results<-list(T=T, MultiPvalue=MultiPvalue,NRVT=NRVT,NRVT_Num=NRVT_Num,
                    EmpiricalTetrads=EmpiricalTetrads,Modelcov=Modelcov)
return(results)
}
########### Tetrad Function End#######################################################
```

```
######################################################################################
################################### Nested or not Function ##############################
######################################################################################
### Information needed  ###
### 1. Two sample covariance matrices "Samplecov1" and "Samplecov2". ###
### 2. Two SEM models "FactorModel1" and "FactorModel2". ###
### 3. Two sample size "Size1" and "Size2". ###
### 4. Function “TetradAnalysisNoRandom” ###
######################################################################################
NestedOrNot<-function(Samplecov1,FactorModel1,Size1,Samplecov2,FactorModel2,Size2){
	Model1<-TetradAnalysisNoRandom(Samplecov1,FactorModel1,Size1)
	Model2<-TetradAnalysisNoRandom(Samplecov2,FactorModel2,Size2)
	EmpiricalTetrads1<-Model1$EmpiricalTetrads
	EmpiricalTetrads2<-Model2$EmpiricalTetrads

	if (nrow(EmpiricalTetrads1)<=nrow(EmpiricalTetrads2)){		# Check which model has more
vanishing tetrads after empirical method. #
	if (all(EmpiricalTetrads1[,1] %in% EmpiricalTetrads2[,1])==TRUE){		# If all vanishing tetrads
for one model are all exist in another model, then they are nested. #
		nest<-"Two models are nested"}else{
		nest<-"Two models are not nested"}
	}else{
		if (all(EmpiricalTetrads2[,1] %in% EmpiricalTetrads1[,1])==TRUE){
		nest<-"Two models are nested"}else{
		nest<-"Two models are not nested"}
	}
	results<-list(Model3BOV=Model1$Modelcov,
			Model2COV=Model2$Modelcov,
			NRVT1=Model1$NRVT,
			NRVT2=Model2$NRVT,
			NRVT_Num1=Model1$NRVT_Num,
			NRVT_Num2=Model2$NRVT_Num,
			NEST=nest)
### Output nonredundant tetrad setting, nonredundant tetrad number and nested or not status for
these two models. ###

	return(results)
}
########### NestedOrNot Function End###############################################
```

```
##################################################################################
############################### TetradAnalysisOnly  Function ##########################
##################################################################################
### Information needed ###
### 1. Nonredundant vanishing tetrad setting: “NRVT”. ###
### 2. Nonredundant vanishing tetrad number: “NRVT_Num”.###
### 3. Sample covariance/correlation matrix " Samplecov ". ###

######### Individual Test ###########################################################
TetradAnalysisOnly<-function(Samplecov, NRVT, NRVT_Num) {
Tetrad<-matrix(1,nc=1,nr=NRVT_Num)
AVAR<-matrix(1,nc=1,nr=NRVT_Num)
Teststat<-matrix(1,nc=1,nr=NRVT_Num)
Pvalue<-matrix(1,nc=1,nr=NRVT_Num)
Fun<-t(t(NRVT))
NRVT_COV<-matrix(0, nrow=NRVT_Num,ncol=4)
for (k in 1:NRVT_Num){
        e<-Fun[k,1]%/%1000
        f<-Fun[k,1]%/%100%%10
        g<-Fun[k,1]%/%10%%10
        h<-Fun[k,1]%%10
    sigma<-
c(min(e,f)*10+max(e,f),min(g,h)*10+max(g,h),min(e,g)*10+max(e,g),min(f,h)*10+max(f,h))
    NRVT_COV[k,1]<-sigma[1]
    NRVT_COV[k,2]<-sigma[2]
    NRVT_COV[k,3]<-sigma[3]
    NRVT_COV[k,4]<-sigma[4]
    SigmaSS<-matrix(1,nc=length(sigma),nr=length(sigma))
        for (i in 1:length(sigma)){
            for (j in 1:length(sigma)) {
            ee<-sigma[i]%/%10
            ff<-sigma[i]%%10
            gg<-sigma[j]%/%10
            hh<-sigma[j]%%10
            efgh<-Samplecov[ee,gg]*Samplecov[ff,hh]+Samplecov[ee,hh]*Samplecov[ff,gg]
        SigmaSS[i,j]<-efgh
            }
        }
dt<-matrix(c(Samplecov[g,h],Samplecov[e,f],-Samplecov[f,h],-Samplecov[e,g]),ncol=4,nrow=1)
VAR<-(1/Size)* dt %*% SigmaSS %*% t(dt)
T<-Samplecov[e,f]*Samplecov[g,h]-Samplecov[e,g]*Samplecov[f,h]
stat<-T^2/VAR
Chipvalue<-1-pchisq(stat,df=1)
    Tetrad[k,1]<-round(T,3)
    AVAR[k,1]<-round(VAR,3)
    Teststat[k,1]<-round(stat,3)
    Pvalue[k,1]<-round(Chipvalue,3)
}
Result<-cbind(Fun,Tetrad,AVAR,Teststat,Pvalue)
colnames(Result) <- c("Model Implied Tetrad","t","AVAR","TestStatistic","P-value")

######### Multivariate test ####################################################
uc<-unique(as.vector(NRVT_COV))
    SigmaSS<-matrix(1,nc=length(uc),nr=length(uc))
        for (i in 1:length(uc)){
            for (j in 1:length(uc)) {
            e<-uc[i]%/%10
            f<-uc[i]%%10
            g<-uc[j]%/%10
            h<-uc[j]%%10
            efgh<-Samplecov[e,g]*Samplecov[f,h]+Samplecov[e,h]*Samplecov[f,g]
            SigmaSS[i,j]<-efgh
                }
            }
    dtaudsigmaCOV<-matrix(0,ncol=length(uc),nrow=NRVT_Num)
        for (i in 1:nrow(dtaudsigmaCOV)){
            for(j in 1:ncol(dtaudsigmaCOV)){
            if (NRVT_COV[i,1]==uc[j])
{dtaudsigmaCOV[i,j]=Samplecov[NRVT_COV[i,2]%/%10,NRVT_COV[i,2]%%10]}
            if (NRVT_COV[i,2]==uc[j])
{dtaudsigmaCOV[i,j]=Samplecov[NRVT_COV[i,1]%/%10,NRVT_COV[i,1]%%10]}
            if (NRVT_COV[i,3]==uc[j]) {dtaudsigmaCOV[i,j]=-
Samplecov[NRVT_COV[i,4]%/%10,NRVT_COV[i,4]%%10]}
            if (NRVT_COV[i,4]==uc[j]) {dtaudsigmaCOV[i,j]=-
Samplecov[NRVT_COV[i,3]%/%10,NRVT_COV[i,3]%%10]}
            }
        }
SigmaTT<-dtaudsigmaCOV %*% SigmaSS %*% t(dtaudsigmaCOV)
t<-t(t(Result[,2]))
T<-Size*t(t) %*% solve(SigmaTT) %*% t
MultiPvalue<-1-pchisq(T,df=NRVT_Num)
colnames(T) <- c("TestStatistic For Multivariate Test")
colnames(MultiPvalue) <- c("P-value For Multivariate Test")
results<-list(T=T, MultiPvalue=MultiPvalue)
return(results)
}
########### TetradAnalysisOnly  Function End#############################################
```

APPENDIX B

R CODES FOR GENERATING UNIFORM RANDOM CORRELATION MATRICES

```
###############################################################################################
############################## UCM method ###############################################
###############################################################################################
set.seed(123)                     # Random seed for data generation
d<-6                              # Matrix dimension
UCM<-list()                       # Create a list to keep generated data
m<-1
k<-10000                          # Number of matrices needed
repeat{
rr <- matrix(0, d, d)
        for (i in 1:d) {
        for (j in i:d){
        rr[i, j] <- runif(1, 0, 1)      # Every correlation is uniform(0,1) distributed #
        rr[j, i]<-rr[i, j]
        }
        }
diag(rr) <- 1
eigen<-eigen(rr)$values
z<-sign(eigen)
if(sum(z)==d){UCM[[m]]<-rr}       # Check eigen values to know a matrix is positive definite or not #
if(sum(z)==d){m<-m+1}
  if(m==k+1){
    break
  }
}
########## “UCM” is a list with 10000 6*6 correlation matrices ##########
```

```
#########################################################################################
################# Partial Correlation Method ############################################
#########################################################################################

# Function 'PCM' created for random correlation method #####
PCM<-function (d, alphad = 1)                # d is the dimension of matrix,
{                                            # alphad=1 means we set correlation distribution Beta(1,1)
  rjm<-function (rsub, alp)
 {
  b <- nrow(rsub)
  ii <- 2:(b - 1)
  r1 <- rsub[ii, 1]
  r3 <- rsub[ii, b]
  R2 <- rsub[ii, ii]
  Ri <- solve(R2)
  rcond <- rbeta(1, alp, alp)                # Beta(1,1) is the same as Uniform(0,1)
  tem13 <- t(r1) %*% Ri %*% r3
  tem11 <- t(r1) %*% Ri %*% r1
  tem33 <- t(r3) %*% Ri %*% r3
  res <- tem13 + rcond * sqrt((1 - tem11) * (1 - tem33))    # give res a random value based on tem13 #
  return(res)
 }
  d <- as.integer(d)
  if (d <= 0 || !is.integer(d)) {
    stop("The dimension 'd' should be a positive integer!\n")
  }
  if (alphad <= 0) {
    stop("'alphad' should be positive!\n")
  }
  if (d == 1) {
    rr <- matrix(1, 1, 1)
    return(rr)
  }
  if (d == 2) {
    rho <- runif(1, 0, 1)
    rr <- matrix(c(1, rho, rho, 1), 2, 2)
    return(rr)
  }
  rr <- matrix(0, d, d)
  diag(rr) <- 1
  for (i in 1:(d - 1)) {
    alp <- alphad + (d - 2)/2
    rr[i, i + 1] <- rbeta(1, alp, alp)
    rr[i + 1, i] <- rr[i, i + 1]
    }
    for (m in 2:(d - 1)) {
        for (j in 1:(d - m)) {
            rsub <- rr[j:(j + m), j:(j + m)]
            alp <- alphad + (d - 1 - m)/2
            rr[j, j + m] <- rjm(rsub, alp)
            rr[j + m, j] <- rr[j, j + m]
        }
    }
    return(rr)
}
########## Generate 10,000 random correlation matrices use PCM ########
set.seed(123)
dim<-6
p<-dim*dim
Randomcor<-list()                      # “Randomcor” has all generated matrices
j<-1
repeat{
random<- PCM (dim,alphad=1)
eigen<-eigen(random)$values
z<-sign(eigen)
if(sum(z)==dim){Randomcor[[j]]<-random}
if(sum(z)==dim){j<-j+1}
  if(j==10001){
    break
  }
}
```

```
#############################################################################
########################## MCMC method ###############################
#############################################################################

### set burn in 100,000, number of matrices=10,000, thin=10 ###
o<-6
nmat<-10000             # Number of matrices
irt<-100000             # Burn in
thin<-irt/nmat          # For all generated matrices, remain 1 matrix in “thin” matrices
nr<-o*(o-1)/2           # number of correlation in matrix
xcand<-rep(0.5,nr)
xtemp<-rep(0,nr)
jmpsize<-0.5            # jump size between 0 and 1, affact the speed
reject<-0
MCMC<-list()            # Final matrices store in “MCMC”
MCMCall<-list()
mm<-1
ii<-1
rejectY<-0
rejectR<-0
n<-0
set.seed(123)
repeat{
        n<-n+1
        tmp<-rnorm(nr,0,1)
        xtemp<-runif(1,0,1)
        xtmp<-rep(0,nr)
        for (i in 1:nr){
                xtmp[i]<-xtemp^(1/nr)*tmp[i]/length(tmp)
                }
        ycand<-rep(0,nr)
        yflag<-rep(1,nr)
        for (i in 1:nr){
                ycand[i]<-xcand[i]+jmpsize*xtmp[i]
                if(ycand[i]>0 && ycand[i]<1){yflag[i]<-0}
                }
        if(sum(yflag)!=0){rejectY<-rejectY+1}
        if(sum(yflag)==0){
        k<-1
        rr <- matrix(1, o, o)
        for (m in 1:(o - 1)) {
    for (j in 1:(o - m)) {
      rr[j, j + m] <- ycand[k]
      rr[j + m, j] <- rr[j, j + m]
```

```
        k<-k+1
    }
        }
    eigen<-eigen(rr)$values          # Check positive definite
    z<-sign(eigen)
    if(sum(z)!=o){rejectR<-rejectR+1}
    if(sum(z)==o){
        xcand<-ycand
        MCMCall[[mm]]<-rr
        mm<-mm+1
    if((mm>irt)&(mm%%thin==0)){MCMC[[ii]]<-rr # Skip burn in & remain 1 matrix in “thin” matrices
          ii<-ii+1}
        }
    }
if(ii==nmat+1){break}
}
```

APPENDIX C

R CODES FOR COMPLEXITY COMPARISON FOR COMPARISON SET 3 IN SECTION 4.4

```
######################################################################################
############################### Comparison 3 ##########################################
######################################################################################

# Roughly generate data based on SEM model, then apply Tetrad function to get Nonredundant tetrads
set and tetrad number ###
############################## Model 3A #########################################
## Simulate data ###
set.seed(123)
x1<-rnorm(n=100,0,1)
x2<-0.8*x1+rnorm(n=100,0,1)
x3<-0.8*x2+rnorm(n=100,0,1)
x4<-0.8*x3+rnorm(n=100,0,1)
X3A<-cbind(as.matrix(x1),as.matrix(x2),as.matrix(x3),as.matrix(x4))
Samplecov3A<-var(X3A)
cor(X3A)
colnames(Samplecov3A) <- c('x1','x2','x3','x4')

### Get tetrad information ###
FactorModel3A<-'
x1 ~ x2
x2 ~ x3
x3 ~ x4
'
Size<-100
Samplecov3A
FactorOut3A<-cfa(FactorModel3A, sample.cov=Samplecov3A, sample.nobs =Size)
summary(FactorOut3A)

# Model degrees of freedom=3, number of unknown parameter =4*5/2-3=7 #

# Model implied tetrad for model 3A #
ExampleTetrad3A<- TetradAnalysisNoRandom (FactorModel2A, Samplecov2A, size)
NRVT3ANoRandom<-1342
NRVT_Num3ANoRandom<-1
#       Tetrads: 1342   #
#       Tetrad number: 1        #
```

```
###################### Model 3B ##############################
### Simulate data ###
set.seed(123)
y1<-rnorm(n=100,0,1)
x1<-Y1+rnorm(n=100,0,1)
x2<-0.8*Y1+rnorm(n=100,0,1)
x3<-0.8*Y1+rnorm(n=100,0,1)
x4<-0.5*Y1+rnorm(n=100,0,1)

X3B<-cbind(as.matrix(x1),as.matrix(x2),as.matrix(x3),as.matrix(x4))
### we can get both correlation matrix and covariance matrix in known model analysis ###
Samplecov3B<-var(X3B)
cor(X3B)
colnames(Samplecov3B) <- c('x1','x2','x3','x4')

### Get tetrad information use same parameters for x2 and x3 ###
FactorModel3B<-'
y1 =~ x1 + v2*x2 + v2*x3 + x4
'
Size<-100
Samplecov3B
FactorOut3B<-cfa(FactorModel3B, sample.cov=Samplecov3B, sample.nobs =Size)
summary(FactorOut3B)

# Model degrees of freedom=3, number of unknown parameter =4*5/2-3=7 #

# Model implied tetrad for model 3B #
ExampleTetrad3B<- TetradAnalysisNoRandom (FactorModel3B, Samplecov3B, size)
NRVT3BNoRandom<-c(1342,1234)
NRVT_Num3BNoRandom<-2

#	All 3 tetrads are implied, use 1342 and pick one from the left two vanishing tetrads	#
#	Tetrads: 1342 1234	#
#	t=2			#
```

```
############# Complexity analysis use uniform random correlation matrices ##################

### Generate random correlation matrices ###

########################################################################
######### MCMC method to generate uniform distributed correlation matrix ######
########################################################################

### set burn in 10000, number of matrices=1000, thin=10 ###
o<-4
nmat<-10000             # Number of matrices
irt<-100000
thin<-irt/nmat
nr<-o*(o-1)/2
xcand<-rep(0.5,nr)
xtemp<-rep(0,nr)
jmpsize<-0.5
reject<-0
MCMC<-list()
MCMCall<-list()
mm<-1
ii<-1
rejectY<-0
rejectR<-0
n<-0
set.seed(123)
repeat{
        n<-n+1
        tmp<-rnorm(nr,0,1)
        xtemp<-runif(1,0,1)
        xtmp<-rep(0,nr)
        for (i in 1:nr){
                xtmp[i]<-xtemp^(1/nr)*tmp[i]/length(tmp)
                }
        ycand<-rep(0,nr)
        yflag<-rep(1,nr)
        for (i in 1:nr){
                ycand[i]<-xcand[i]+jmpsize*xtmp[i]
                if(ycand[i]>0 && ycand[i]<1){yflag[i]<-0}
                }
        if(sum(yflag)!=0){rejectY<-rejectY+1}
        if(sum(yflag)==0){
        k<-1
        rr <- matrix(1, o, o)
        for (m in 1:(o - 1)) {
    for (j in 1:(o - m)) {
       rr[j, j + m] <- ycand[k]
       rr[j + m, j] <- rr[j, j + m]
                k<-k+1
        }
                }
        eigen<-eigen(rr)$values
        z<-sign(eigen)
        if(sum(z)!=o){rejectR<-rejectR+1}
        if(sum(z)==o){
                xcand<-ycand
                MCMCall[[mm]]<-rr
                mm<-mm+1
        if((mm>irt)&(mm%%thin==0)){
                colnames(rr) <- c('x1','x2','x3','x4')
                MCMC[[ii]]<-rr
                ii<-ii+1
                        }
                }
        }
if(ii==nmat+1){break}
}

### List “MCMC” contains 10,000 matrices ###
```

```
##################### Fitting results for Model 3A #################################
PResultsA<-numeric(10000)
TResultsA<-numeric(10000)
TestsA<-numeric(10000)
TRMSEAA<-numeric(10000)
SRMRA<-numeric(10000)
ChisquareA<-numeric(10000)
RMSEAA<-numeric(10000)
AICA<-numeric(10000)
BICA<-numeric(10000)
PvalueA<-numeric(10000)
GFIA<-numeric(10000)
AGFIA<-numeric(10000)
CFIA<-numeric(10000)
TLIA<-numeric(10000)
ConvergedA<-numeric(10000)

for (i in 1:10000){
        Samplecor<-MCMC[[i]]
        colnames(Samplecor) <- c('x1','x2','x3','x4')
        x<-TetradAnalysisOnly(Samplecor,NRVT3ANoRandom,NRVT_Num3ANoRandom)
        MultiPvalue<-x$MultiPvalue
        PResultsA[i]<-x$MultiPvalue
        TResultsA[i]<-x$T
        TRMSEAA[i]<-sqrt(max(0,x$T-NRVT_Num2ANoRandom))/sqrt(NRVT_Num2ANoRandom*(100-
1))
        if(MultiPvalue<0.05){TestsA[i]<-1} else{TestsA[i]<-0}

        FactorOut3A<-cfa(FactorModel3A, sample.cov=Samplecor, sample.nobs =Size)
        ConvergedA[i]<-lavInspect(FactorOut3A, what = "converged")
        if (ConvergedA[i]==0){
                        SRMRA[i]<-0
                        ChisquareA[i]<-0
                        RMSEAA[i]<-0
                        AICA[i]<-0
                        BICA[i]<-0
                        PvalueA[i]<-0
                        GFIA[i]<-0
                        CFIA[i]<-0
                        TLIA[i]<-0
                        AGFIA[i]<-0
                        }
                else{
                        SRMRA[i]<-fitMeasures(FactorOut2A, "srmr")
                        ChisquareA[i]<-fitMeasures(FactorOut3A, "chisq")
                        RMSEAA[i]<-fitMeasures(FactorOut3A, "rmsea")
                        AICA[i]<-fitMeasures(FactorOut3A, "aic")
                        BICA[i]<-fitMeasures(FactorOut3A, "bic")
                        PvalueA[i]<-fitMeasures(FactorOut3A, "pvalue")
                        GFIA[i]<-fitMeasures(FactorOut3A, "gfi")
                        AGFIA[i]<-fitMeasures(FactorOut3A, "agfi")
                        CFIA[i]<-fitMeasures(FactorOut3A, "cfi")
                        TLIA[i]<-fitMeasures(FactorOut3A, "tli")
                        }
}

setwd("C:/Users/Administrator/Desktop")
ResultsA <-
data.frame(ChisquareA,PvalueA,SRMRA,RMSEAA,GFIA,AGFIA,CFIA,TLIA,AICA,BICA,TResultsA,PResultsA,T
RMSEAA,ConvergedA)
write.csv (x =ResultsA , file = "Results3ASeed123N10000.csv")


##################### Fitting results for Model 3B #################################
PResultsB<-numeric(10000)
TResultsB<-numeric(10000)
TestsB<-numeric(10000)
TRMSEAB<-numeric(10000)
SRMRB<-numeric(10000)
ChisquareB<-numeric(10000)
RMSEAB<-numeric(10000)
AICB<-numeric(10000)
BICB<-numeric(10000)
PvalueB<-numeric(10000)
GFIB<-numeric(10000)
AGFIB<-numeric(10000)
CFIB<-numeric(10000)
TLIB<-numeric(10000)
ConvergedB<-numeric(10000)

for (i in 1:10000){
        Samplecor<-MCMC[[i]]
        colnames(Samplecor) <- c('x1','x2','x3','x4')
        x<-TetradAnalysisOnly(Samplecor,NRVT3BNoRandom,NRVT_Num3BNoRandom)
        MultiPvalue<-x$MultiPvalue
        PResultsB[i]<-x$MultiPvalue
        TResultsB[i]<-x$T
```

```
    TRMSEAB[i]<-sqrt(max(0,x$T-NRVT_Num2BNoRandom))/sqrt(NRVT_Num2BNoRandom*(100-
1))
    if(MultiPvalue<0.05){TestsB[i]<-1} else{TestsB[i]<-0}

    FactorOut2B<-cfa(FactorModel3B, sample.cov=Samplecor, sample.nobs =Size)
    ConvergedB[i]<-lavInspect(FactorOut3B, what = "converged")
    if (ConvergedB[i]==0){
            SRMRB[i]<-0
            ChisquareB[i]<-0
            RMSEAB[i]<-0
            AICB[i]<-0
            BICB[i]<-0
            PvalueB[i]<-0
            GFIB[i]<-0
            CFIB[i]<-0
            TLIB[i]<-0
            AGFIB[i]<-0
            }
        else{
            SRMRB[i]<-fitMeasures(FactorOut3B, "srmr")
            ChisquareB[i]<-fitMeasures(FactorOut3B, "chisq")
            RMSEAB[i]<-fitMeasures(FactorOut3B, "rmsea")
            AICB[i]<-fitMeasures(FactorOut3B, "aic")
            BICB[i]<-fitMeasures(FactorOut3B, "bic")
            PvalueB[i]<-fitMeasures(FactorOut3B, "pvalue")
            GFIB[i]<-fitMeasures(FactorOut3B, "gfi")
            AGFIB[i]<-fitMeasures(FactorOut3B, "agfi")
            CFIB[i]<-fitMeasures(FactorOut3B, "cfi")
            TLIB[i]<-fitMeasures(FactorOut3B, "tli")
            }
}
ResultsB <-
data.frame(ChisquareB,PvalueB,SRMRB,RMSEAB,GFIB,AGFIB,CFIB,TLIB,AICB,BICB,TResultsB,PResultsB,T
RMSEAB,ConvergedB)
write.csv (x =ResultsB , file = "Results3BSeed123N10000.csv")
```

```
### Generate random population correlation matrices from known model ###

################## From Model 3A #############################################
# Set all observed variables have variance equal to 1, sample size equal to 100
Size<-100
set.seed(123)
j<-0
beta1<-c()
beta2<-c()
beta3<-c()
KNOWN3A<-list()
for (i in 1:10000){
	beta1[i]<-round(runif(1,0.05,0.95),3)
	beta2[i]<-round(runif(1,0.05,0.95),3)
	beta3[i]<-round(runif(1,0.05,0.95),3)
	cov12<-beta1[i]
	cov13<-beta1[i]*beta2[i]
	cov14<-beta1[i]*beta2[i]*beta3[i]
	cov23<-beta2[i]
	cov24<-beta2[i]*beta3[i]
	cov34<-beta3[i]
	CorMatrix <- matrix(rep(1,4*4), nrow=4)
	CorMatrix[1,2]<-CorMatrix[2,1]<-cov12
	CorMatrix[1,3]<-CorMatrix[3,1]<-cov13
	CorMatrix[1,4]<-CorMatrix[4,1]<-cov14
	CorMatrix[2,3]<-CorMatrix[3,2]<-cov23
	CorMatrix[2,4]<-CorMatrix[4,2]<-cov24
	CorMatrix[3,4]<-CorMatrix[4,3]<-cov34
	if(is.positive.definite(CorMatrix, tol=1e-8)=="TRUE"){
	KNOWN3A[[j+1]]<-CorMatrix
	j<-j+1
	}
}

# List “KNOWN3A” contain 10,000 random population correlation matrices generated from Model 3A #
```

```
################## From Model 3B #############################################
j<-0
Size<-100
set.seed(123)
KNOWN3B<-list()
beta1<-c()
beta2<-c()
beta3<-c()
beta4<-c()
for (i in 1:10000){
        beta1[i]<-1
        beta2[i]<-beta3[i]<-round(runif(1,0.05,0.95),3)
        beta4[i]<-round(runif(1,0.05,0.95),3)
        cov12<-beta2[i]
        cov13<-beta3[i]
        cov14<-beta4[i]
        cov23<-beta2[i]*beta3[i]
        cov24<-beta2[i]*beta4[i]
        cov34<-beta3[i]*beta4[i]
        CorMatrix <- matrix(rep(1,4*4), nrow=4)
        CorMatrix[1,2]<-CorMatrix[2,1]<-cov12
        CorMatrix[1,3]<-CorMatrix[3,1]<-cov13
        CorMatrix[1,4]<-CorMatrix[4,1]<-cov14
        CorMatrix[2,3]<-CorMatrix[3,2]<-cov23
        CorMatrix[2,4]<-CorMatrix[4,2]<-cov24
        CorMatrix[3,4]<-CorMatrix[4,3]<-cov34
        if(is.positive.definite(CorMatrix, tol=1e-8)=="TRUE"){
        KNOWN3B[[j+1]]<-CorMatrix
        j<-j+1
        }
}

# List “KNOWN3B” contain 10,000 random population correlation matrices generated from Model 3A #
```

```
###################### Fitting results for matrices from known models ################
# Fitting results for Model 3A in matrices generated from Model 3B #
PResultsA<-numeric(10000)
TResultsA<-numeric(10000)
TestsA<-numeric(10000)
TRMSEAA<-numeric(10000)
SRMRA<-numeric(10000)
ChisquareA<-numeric(10000)
RMSEAA<-numeric(10000)
AICA<-numeric(10000)
BICA<-numeric(10000)
PvalueA<-numeric(10000)
GFIA<-numeric(10000)
AGFIA<-numeric(10000)
CFIA<-numeric(10000)
TLIA<-numeric(10000)
ConvergedA<-numeric(10000)

for (i in 1:10000){
	Samplecor<-KNOWN3B[[i]]
	colnames(Samplecor) <- c('x1','x2','x3','x4')
	x<-TetradAnalysisOnly(Samplecor,NRVT3ANoRandom,NRVT_Num3ANoRandom)
	MultiPvalue<-x$MultiPvalue
	PResultsA[i]<-x$MultiPvalue
	TResultsA[i]<-x$T
	TRMSEAA[i]<-sqrt(max(0,x$T-NRVT_Num3ANoRandom))/sqrt(NRVT_Num3ANoRandom*(100-
1))
	if(MultiPvalue<0.05){TestsA[i]<-1} else{TestsA[i]<-0}
	FactorOutA<-cfa(FactorModel3A, sample.cov=Samplecor, sample.nobs =Size)
	ConvergedA[i]<-lavInspect(FactorOutA, what = "converged")
	if (ConvergedA[i]==1){
			SRMRA[i]<-fitMeasures(FactorOutA, "srmr")
			ChisquareA[i]<-fitMeasures(FactorOutA, "chisq")
			RMSEAA[i]<-fitMeasures(FactorOutA, "rmsea")
			AICA[i]<-fitMeasures(FactorOutA, "aic")
			BICA[i]<-fitMeasures(FactorOutA, "bic")
			PvalueA[i]<-fitMeasures(FactorOutA, "pvalue")
			GFIA[i]<-fitMeasures(FactorOutA, "gfi")
			AGFIA[i]<-fitMeasures(FactorOutA, "agfi")
			CFIA[i]<-fitMeasures(FactorOutA, "cfi")
			TLIA[i]<-fitMeasures(FactorOutA, "tli")
			}
		else{
			SRMRA[i]<-0
```

```
                              ChisquareA[i]<-0
                              RMSEAA[i]<-0
                              AICA[i]<-0
                              BICA[i]<-0
                              PvalueA[i]<-0
                              GFIA[i]<-0
                              CFIA[i]<-0
                              TLIA[i]<-0
                              AGFIA[i]<-0
                              }
}

setwd("C:/Users/Administrator/Desktop")
ResultsA <-
data.frame(ChisquareA,PvalueA,SRMRA,RMSEAA,GFIA,AGFIA,CFIA,TLIA,AICA,BICA,TResultsA,PResultsA,T
RMSEAA,ConvergedA)
write.csv (x =ResultsA , file = "Known3BResults3AN10000.csv")
```

```
# Fitting results for Model 3B in matrices generated from Model 3A #

PResultsB<-numeric(10000)
TResultsB<-numeric(10000)
TestsB<-numeric(10000)
TRMSEAB<-numeric(10000)
SRMRB<-numeric(10000)
ChisquareB<-numeric(10000)
RMSEAB<-numeric(10000)
AICB<-numeric(10000)
BICB<-numeric(10000)
PvalueB<-numeric(10000)
GFIB<-numeric(10000)
AGFIB<-numeric(10000)
CFIB<-numeric(10000)
TLIB<-numeric(10000)
ConvergedB<-numeric(10000)

for (i in 1:10000){
        Samplecor<-KNOWN3A[[i]]
        colnames(Samplecor) <- c('x1','x2','x3','x4')
        x<-TetradAnalysisOnly(Samplecor,NRVT3BNoRandom,NRVT_Num3BNoRandom)
        MultiPvalue<-x$MultiPvalue
        PResultsB[i]<-x$MultiPvalue
        TResultsB[i]<-x$T
        TRMSEAB[i]<-sqrt(max(0,x$T-NRVT_Num3BNoRandom))/sqrt(NRVT_Num3BNoRandom*(100-1))
        if(MultiPvalue<0.05){TestsB[i]<-1} else{TestsB[i]<-0}

        FactorOutB<-cfa(FactorModel3B, sample.cov=Samplecor, sample.nobs =Size)
        ConvergedB[i]<-lavInspect(FactorOutB, what = "converged")

        if (ConvergedB[i]==1){
                        SRMRB[i]<-fitMeasures(FactorOutB, "srmr")
                        ChisquareB[i]<-fitMeasures(FactorOutB, "chisq")
                        RMSEAB[i]<-fitMeasures(FactorOutB, "rmsea")
                        AICB[i]<-fitMeasures(FactorOutB, "aic")
                        BICB[i]<-fitMeasures(FactorOutB, "bic")
                        PvalueB[i]<-fitMeasures(FactorOutB, "pvalue")
                        GFIB[i]<-fitMeasures(FactorOutB, "gfi")
                        AGFIB[i]<-fitMeasures(FactorOutB, "agfi")
                        CFIB[i]<-fitMeasures(FactorOutB, "cfi")
                        TLIB[i]<-fitMeasures(FactorOutB, "tli")
                        }
```

```
                    else{
                              SRMRB[i]<-0
                              ChisquareB[i]<-0
                              RMSEAB[i]<-0
                              AICB[i]<-0
                              BICB[i]<-0
                              PvalueB[i]<-0
                              GFIB[i]<-0
                              CFIB[i]<-0
                              TLIB[i]<-0
                              AGFIB[i]<-0
                              }
}
ResultsB <-
data.frame(ChisquareB,PvalueB,SRMRB,RMSEAB,GFIB,AGFIB,CFIB,TLIB,AICB,BICB,TResultsB,PResultsB,T
RMSEAB,ConvergedB)
write.csv (x =ResultsB , file = "Known3AResults3BN10000.csv")
```

```
####################### R codes for comparison tables and plots ####################
############ 3A vs 3B when random seed is 123, N=10,000 ###########
ResultsAData <- read.csv(file=" Results3ASeed123N10000.csv", header=TRUE, sep=",")
ResultsBData <- read.csv(file=" Results3BSeed123N10000.csv", header=TRUE, sep=",")

ChisquareA<-ResultsAData["ChisquareA"][1:10000,]
PvalueA <-ResultsAData["PvalueA"][1:10000,]
SRMRA<-ResultsAData["SRMRA"][1:10000,]
RMSEAA<-ResultsAData["RMSEAA"][1:10000,]
GFIA<-ResultsAData["GFIA"][1:10000,]
AGFIA<-ResultsAData["AGFIA"][1:10000,]
CFIA<-ResultsAData["CFIA"][1:10000,]
TLIA<-ResultsAData["TLIA"][1:10000,]
AICA<-ResultsAData["AICA"][1:10000,]
BICA<-ResultsAData["BICA"][1:10000,]
TResultsA<-ResultsAData["TResultsA"][1:10000,]
PResultsA<-ResultsAData["PResultsA"][1:10000,]
TRMSEAA<-ResultsAData["TRMSEAA"][1:10000,]
ConvergedA<-ResultsAData["ConvergedA"][1:10000,]

ChisquareB<-ResultsBData["ChisquareB"][1:10000,]
PvalueB <-ResultsBData["PvalueB"][1:10000,]
SRMRB<-ResultsBData["SRMRB"][1:10000,]
RMSEAB<-ResultsBData["RMSEAB"][1:10000,]
GFIB<-ResultsBData["GFIB"][1:10000,]
AGFIB<-ResultsBData["AGFIB"][1:10000,]
CFIB<-ResultsBData["CFIB"][1:10000,]
TLIB<-ResultsBData["TLIB"][1:10000,]
AICB<-ResultsBData["AICB"][1:10000,]
BICB<-ResultsBData["BICB"][1:10000,]
TResultsB<-ResultsBData["TResultsB"][1:10000,]
PResultsB<-ResultsBData["PResultsB"][1:10000,]
TRMSEAB<-ResultsBData["TRMSEAB"][1:10000,]
ConvergedB<-ResultsBData["ConvergedB"][1:10000,]
```

```
### Comparison results ###
# Chisquare
# Get the values when 3A and 3B both converged #
Chisquare<-cbind(ChisquareA,ChisquareB,ConvergedA,ConvergedB)
ChisquareValid<-Chisquare[Chisquare[,3]==1& Chisquare[,4]==1,]
ChisquareMean<-cbind(mean(ChisquareValid[,1]),mean(ChisquareValid[,2]))
ChisquareVar<-cbind(var(ChisquareValid[,1]),var(ChisquareValid[,2]))
ChisquareABetter<-nrow(ChisquareValid[ChisquareValid[,1]<ChisquareValid[,2],])
ChisquareBBetter<-nrow(ChisquareValid[ChisquareValid[,1]>ChisquareValid[,2],])
ChisquareOut<-
list(ChisquareMean=ChisquareMean,ChisquareVar=ChisquareVar,ChisquareABetter=ChisquareABetter,ChisquareBBetter=ChisquareBBetter)

# Pvalue
Pvalue<-cbind(PvalueA,PvalueB,ConvergedA,ConvergedB)
PvalueValid<-Pvalue[Pvalue[,3]==1& Pvalue[,4]==1,]
PvalueMean<-cbind(mean(PvalueValid[,1]),mean(PvalueValid[,2]))
PvalueVar<-cbind(var(PvalueValid[,1]),var(PvalueValid[,2]))
PvalueABetter<-nrow(PvalueValid[PvalueValid[,1]>PvalueValid[,2],])
PvalueBBetter<-nrow(PvalueValid[PvalueValid[,1]<PvalueValid[,2],])
PvalueNoBetter<-nrow(PvalueValid[PvalueValid[,1]==PvalueValid[,2],])
PvalueA0.05<-nrow(PvalueValid[PvalueValid[,1]>0.05,])
PvalueB0.05<-nrow(PvalueValid[PvalueValid[,2]>0.05,])
PvalueAB0.05<-nrow(PvalueValid[PvalueValid[,1]>0.05 &PvalueValid[,2]>0.05,])
PvalueOut<-
list(PvalueMean=PvalueMean,PvalueVar=PvalueVar,PvalueABetter=PvalueABetter,PvalueBBetter=PvalueBBetter,PvalueNoBetter=PvalueNoBetter,PvalueA0.05=PvalueA0.05,PvalueB0.05=PvalueB0.05,PvalueAB0.05=PvalueAB0.05)

# SRMR
SRMR<-cbind(SRMRA,SRMRB,ConvergedA,ConvergedB)
SRMRValid<-SRMR[SRMR[,3]==1& SRMR[,4]==1,]
SRMRMean<-cbind(mean(SRMRValid[,1]),mean(SRMRValid[,2]))
SRMRVar<-cbind(var(SRMRValid[,1]),var(SRMRValid[,2]))
SRMRABetter<-nrow(SRMRValid[SRMRValid[,1]<SRMRValid[,2],])
SRMRBBetter<-nrow(SRMRValid[SRMRValid[,1]>SRMRValid[,2],])
SRMRNoBetter<-nrow(SRMRValid[SRMRValid[,1]==SRMRValid[,2],])
SRMRA0.08<-nrow(SRMRValid[SRMRValid[,1]<0.08,])
SRMRB0.08<-nrow(SRMRValid[SRMRValid[,2]<0.08,])
SRMRAB0.08<-nrow(SRMRValid[SRMRValid[,1]<0.08 &SRMRValid[,2]<0.08,])
SRMROut<-
list(SRMRMean=SRMRMean,SRMRVar=SRMRVar,SRMRABetter=SRMRABetter,SRMRBBetter=SRMRBBetter,SRMRNoBetter=SRMRNoBetter,SRMRA0.08=SRMRA0.08,SRMRB0.08=SRMRB0.08,SRMRAB0.08=SRMRAB0.08)
```

```
# RMSEA
RMSEA<-cbind(RMSEAA,RMSEAB,ConvergedA,ConvergedB)
RMSEAValid<-RMSEA[RMSEA[,3]==1& RMSEA[,4]==1,]
RMSEAMean<-cbind(mean(RMSEAValid[,1]),mean(RMSEAValid[,2]))
RMSEAVar<-cbind(var(RMSEAValid[,1]),var(RMSEAValid[,2]))
RMSEAABetter<-nrow(RMSEAValid[RMSEAValid[,1]<RMSEAValid[,2],])
RMSEABBetter<-nrow(RMSEAValid[RMSEAValid[,1]>RMSEAValid[,2],])
RMSEANoBetter<-nrow(RMSEAValid[RMSEAValid[,1]==RMSEAValid[,2],])
RMSEAA0.1<-nrow(RMSEAValid[RMSEAValid[,1]<0.1,])
RMSEAB0.1<-nrow(RMSEAValid[RMSEAValid[,2]<0.1,])
RMSEAOut<-
list(RMSEAMean=RMSEAMean,RMSEAVar=RMSEAVar,RMSEAABetter=RMSEAABetter,RMSEABBetter=R
MSEABBetter,RMSEANoBetter=RMSEANoBetter,RMSEAA0.1=RMSEAA0.1,RMSEAB0.1=RMSEAB0.1)

# GFI
GFI<-cbind(GFIA,GFIB,ConvergedA,ConvergedB)
GFIValid<-GFI[GFI[,3]==1& GFI[,4]==1,]
GFIMean<-cbind(mean(GFIValid[,1]),mean(GFIValid[,2]))
GFIVar<-cbind(var(GFIValid[,1]),var(GFIValid[,2]))
GFIABetter<-nrow(GFIValid[GFIValid[,1]>GFIValid[,2],])
GFIBBetter<-nrow(GFIValid[GFIValid[,1]<GFIValid[,2],])
GFINoBetter<-nrow(GFIValid[GFIValid[,1]==GFIValid[,2],])
GFIA0.95<-nrow(GFIValid[GFIValid[,1]>0.95,])
GFIB0.95<-nrow(GFIValid[GFIValid[,2]>0.95,])
GFIOut<-list(GFIMean=GFIMean,GFIVar=GFIVar,GFIABetter=GFIABetter,GFIBBetter=GFIBBetter,
                    GFINoBetter=GFINoBetter,GFIA0.95=GFIA0.95,GFIB0.95=GFIB0.95)

# AGFI
AGFI<-cbind(AGFIA,AGFIB,ConvergedA,ConvergedB)
AGFIValid<-AGFI[AGFI[,3]==1& AGFI[,4]==1,]
AGFIMean<-cbind(mean(AGFIValid[,1]),mean(AGFIValid[,2]))
AGFIVar<-cbind(var(AGFIValid[,1]),var(AGFIValid[,2]))
AGFIABetter<-nrow(AGFIValid[AGFIValid[,1]>AGFIValid[,2],])
AGFIBBetter<-nrow(AGFIValid[AGFIValid[,1]<AGFIValid[,2],])
AGFINoBetter<-nrow(AGFIValid[AGFIValid[,1]==AGFIValid[,2],])
AGFIA0.95<-nrow(AGFIValid[AGFIValid[,1]>0.95,])
AGFIB0.95<-nrow(AGFIValid[AGFIValid[,2]>0.95,])
AGFIOut<-
list(AGFIMean=AGFIMean,AGFIVar=AGFIVar,AGFIABetter=AGFIABetter,AGFIBBetter=AGFIBBetter,
                    AGFINoBetter=AGFINoBetter,AGFIA0.95=AGFIA0.95,AGFIB0.95=AGFIB0.95)
```

```
# CFI
CFI<-cbind(CFIA,CFIB,ConvergedA,ConvergedB)
CFIValid<-CFI[CFI[,3]==1& CFI[,4]==1,]
CFIMean<-cbind(mean(CFIValid[,1]),mean(CFIValid[,2]))
CFIVar<-cbind(var(CFIValid[,1]),var(CFIValid[,2]))
CFIABetter<-nrow(CFIValid[CFIValid[,1]>CFIValid[,2],])
CFIBBetter<-nrow(CFIValid[CFIValid[,1]<CFIValid[,2],])
CFINoBetter<-nrow(CFIValid[CFIValid[,1]==CFIValid[,2],])
CFIA0.95<-nrow(CFIValid[CFIValid[,1]>0.95,])
CFIB0.95<-nrow(CFIValid[CFIValid[,2]>0.95,])
CFIOut<-list(CFIMean=CFIMean,CFIVar=CFIVar,CFIABetter=CFIABetter,CFIBBetter=CFIBBetter,
                    CFINoBetter=CFINoBetter,CFIA0.95=CFIA0.95,CFIB0.95=CFIB0.95)

# TLI
TLI<-cbind(TLIA,TLIB,ConvergedA,ConvergedB)
TLIValid<-TLI[TLI[,3]==1& TLI[,4]==1,]
TLIMean<-cbind(mean(TLIValid[,1]),mean(TLIValid[,2]))
TLIVar<-cbind(var(TLIValid[,1]),var(TLIValid[,2]))
TLIABetter<-nrow(TLIValid[TLIValid[,1]>TLIValid[,2],])
TLIBBetter<-nrow(TLIValid[TLIValid[,1]<TLIValid[,2],])
TLINoBetter<-nrow(TLIValid[TLIValid[,1]==TLIValid[,2],])
TLIA0.95<-nrow(TLIValid[CFIValid[,1]>0.95,])
TLIB0.95<-nrow(TLIValid[CFIValid[,2]>0.95,])
TLIOut<-list(TLIMean=TLIMean,TLIVar=TLIVar,TLIABetter=TLIABetter,TLIBBetter=TLIBBetter,
                    TLINoBetter=TLINoBetter,TLIA0.95=TLIA0.95,TLIB0.95=CFIB0.95)

# AIC
AIC<-cbind(AICA,AICB,ConvergedA,ConvergedB)
AICValid<-AIC[AIC[,3]==1& AIC[,4]==1,]
AICMean<-cbind(mean(AICValid[,1]),mean(AICValid[,2]))
AICVar<-cbind(var(AICValid[,1]),var(AICValid[,2]))
AICABetter<-nrow(AICValid[AICValid[,1]<AICValid[,2],])
AICBBetter<-nrow(AICValid[AICValid[,1]>AICValid[,2],])
AICNoBetter<-nrow(AICValid[AICValid[,1]==AICValid[,2],])
AICOut<-list(AICMean=AICMean,AICVar=AICVar,AICABetter=AICABetter,AICBBetter=AICBBetter,
                    AICNoBetter=AICNoBetter)

# BIC
BIC<-cbind(BICA,BICB,ConvergedA,ConvergedB)
BICValid<-BIC[BIC[,3]==1& BIC[,4]==1,]
BICMean<-cbind(mean(BICValid[,1]),mean(BICValid[,2]))
BICVar<-cbind(var(BICValid[,1]),var(BICValid[,2]))
BICABetter<-nrow(BICValid[BICValid[,1]<BICValid[,2],])
BICBBetter<-nrow(BICValid[BICValid[,1]>BICValid[,2],])
```

```
BICNoBetter<-nrow(BICValid[BICValid[,1]==BICValid[,2],])
BICOut<-list(BICMean=BICMean,BICVar=BICVar,BICABetter=BICABetter,BICBBetter=BICBBetter,
                              BICNoBetter=BICNoBetter)

# TResults
TResults<-cbind(TResultsA,TResultsB)
TResultsMean<-cbind(mean(TResults[,1]),mean(TResults[,2]))
TResultsVar<-cbind(var(TResults[,1]),var(TResults[,2]))
TResultsABetter<-nrow(TResults[TResults[,1]<TResults[,2],])
TResultsBBetter<-nrow(TResults[TResults[,1]>TResults[,2],])
TResultsNoBetter<-nrow(TResults[TResults[,1]==TResults[,2],])

TResultsOut<-list(TResultsMean=TResultsMean,TResultsVar=TResultsVar,TResultsABetter=TResultsABetter,TResultsBBetter=TResultsBBetter, TResultsNoBetter=TResultsNoBetter)

# PResults
PResults<-cbind(PResultsA,PResultsB)
PResultsMean<-cbind(mean(PResults[,1]),mean(PResults[,2]))
PResultsVar<-cbind(var(PResults[,1]),var(PResults[,2]))
PResultsABetter<-nrow(PResults[PResults[,1]>PResults[,2],])
PResultsBBetter<-nrow(PResults[PResults[,1]<PResults[,2],])
PResultsNoBetter<-nrow(PResults[PResults[,1]==PResults[,2],])
PResultsA0.05<-nrow(PResults[PResults[,1]>0.05,])
PResultsB0.05<-nrow(PResults[PResults[,2]>0.05,])
PResultsOut<-list(PResultsMean=PResultsMean,PResultsVar=PResultsVar,PResultsABetter=PResultsABetter,PResultsBBetter=PResultsBBetter,PResultsNoBetter=PResultsNoBetter,PResultsA0.05=PResultsA0.05,PResultsB0.05=PResultsB0.05)

# TRMSEA
TRMSEA<-cbind(TRMSEAA,TRMSEAB)
TRMSEAValid<-TRMSEA[TRMSEA[,1]!=0& TRMSEA[,2]!=0,]
TRMSEAMean<-cbind(mean(TRMSEAValid[,1]),mean(TRMSEAValid[,2]))
TRMSEAVar<-cbind(var(TRMSEAValid[,1]),var(TRMSEAValid[,2]))
TRMSEAABetter<-nrow(TRMSEAValid[TRMSEAValid[,1]<TRMSEAValid[,2],])
TRMSEABBetter<-nrow(TRMSEAValid[TRMSEAValid[,1]>TRMSEAValid[,2],])
TRMSEANoBetter<-nrow(TRMSEAValid[TRMSEAValid[,1]==TRMSEAValid[,2],])
TRMSEAA0.1<-nrow(TRMSEAValid[TRMSEAValid[,1]<0.1,])
TRMSEAB0.1<-nrow(TRMSEAValid[TRMSEAValid[,2]<0.1,])
TRMSEAOut<-list(TRMSEAMean=TRMSEAMean,TRMSEAVar=TRMSEAVar,TRMSEAABetter=TRMSEAABetter,TRMSEABBetter=TRMSEABBetter,TRMSEANoBetter=TRMSEANoBetter,TRMSEAA0.1=TRMSEAA0.1,TRMSEAB0.1=TRMSEAB0.1)
```

```
#TNested
TNested<-TResultsA-TResultsB
mean(TNested[,1])
var(TNested[,1])
Critical<-qchisq(.95, df=1)
TNestedA0.05<-length(TNested[TNested[,1]<Critical,])
1-pchisq(3.84,df=1)
TNestedPvalue<-numeric(10000)
for(i in 1:10000){
	TNestedPvalue[i]<-1-pchisq(TNested[i,1],df=1)}
mean(TNestedPvalue)
var(TNestedPvalue)
sum(TNestedPvalue>=0.05)

##########################################################
# Check how many times SEM converged #
sum(ConvergedA)
sum(ConvergedB)
Bothvalid<-cbind(ConvergedA,ConvergedB)
nrow(Bothvalid[Bothvalid[,1]==1& Bothvalid[,2]==1,])

### Comparison results for Table 4.4.1 ###
ChisquareOut
PvalueOut
SRMROut
RMSEAOut
GFIOut
AGFIOut
CFIOut
TLIOut
AICOut
BICOut

# Tetrad test Results
PResultsOut
nrow(TRMSEA[TRMSEA[,1]!=0,])
nrow(TRMSEA[TRMSEA[,2]!=0,])
nrow(TRMSEA[TRMSEA[,1]!=0& TRMSEA[,2]!=0,])
TRMSEAOut
```

```
####################     R codes for Figure 4.4.3     #########################################
### Pvalue, 0.05 as the cutoff for good fit models ###
par(mfrow=c(2,2))
# Density plot #
densityPvalue3A <- density(PvalueValid[,1])
densityPvalue3B <- density(PvalueValid[,2])
plot(densityPvalue3A ,col='RED', main ='Chi-square test P-value Histogram, Model 3A vs Model
3B',xlab='P-value')
lines(densityPvalue3B,col='BLUE')
abline(v = 0.05,lty=2,col='black')

# CDF plot #
y <- seq(-10, 10, 0.01)
Pvalue3Avalid.ordered = sort(PvalueValid[,1])
n = sum(!is.na(PvalueValid[,1]))
plot(Pvalue3Avalid.ordered , (1:n)/n, ylim = c(0, 1), col='red', type = 's', main = 'Chi-square test P-value
CDFplots, Model 3A vs Model 3B',xlab="P-value")
Pvalue3Bvalid.ordered = sort(PvalueValid[,2])
n = sum(!is.na(PvalueValid[,2]))
lines(Pvalue3Bvalid.ordered ,col='blue',(1:n)/n, type = 's')
abline(h = 0:1, v = 0.05,lty=2,col='black')

### SRMR comparison ###
### SRMR, 0.08 as the cutoff for poor fitting models ###
# Density plot #
DensitySRMR3Avalid <- density(SRMRValid[,1])
DensitySRMR3Bvalid <- density(SRMRValid[,2])
plot(densitySRMR3Avalid   ,col='RED',ylim = c(0, 12),xlim = c(0, 0.4), main ='SRMR Histogram, Model 3A
vs Model 3B',xlab='SRMR value')
lines(densitySRMR3Bvalid   ,col='BLUE')
abline( v = 0.08,lty=2,col='black')

# CDF plot #
y <- seq(-10, 10, 0.01)
SRMR3Avalid.ordered = sort(SRMRValid[,1])
n = sum(!is.na(SRMRValid[,1]))
plot(SRMR3Avalid.ordered , (1:n)/n, ylim = c(0, 1),xlim = c(0, 0.4), col='red', type = 's', main = 'SRMR
CDFplots, Model 3A vs Model 3B',xlab="SRMR value")
SRMR3Bvalid.ordered = sort(SRMRValid[,2])
n = sum(!is.na(SRMRValid[,2]))
lines(SRMR3Bvalid.ordered ,col='blue',(1:n)/n, type = 's')
abline(h = 0:1, v = 0.08,lty=2,col='black')
par(mfrow=c(2,2))
### RMSEA, 0.10 as the cutoff for poor fitting models ###
```

```
# Density plot #
DensityRMSEA3A <- density(RMSEAValid[,1])
DensityRMSEA3B <- density(RMSEAValid[,2])
plot(densityRMSEA3B  ,col='BLUE', ylim = c(0, 4), main ='RMSEA Histogram, Model 3A vs Model
3B',xlab='RMSEA value')
lines(densityRMSEA3A ,col='RED')
abline( v = 0.1,lty=2,col='black')

# CDF plot #
y <- seq(-10, 10, 0.01)
RMSEA3Avalid.ordered = sort(RMSEAValid[,1])
n = sum(!is.na(RMSEAValid[,1]))
plot(RMSEA3Avalid.ordered , (1:n)/n, ylim = c(0, 1), col='red', type = 's', main = 'RMSEA CDFplots, Model
3A vs Model 3B',xlab="RMSEA value")

RMSEA3Bvalid.ordered = sort(RMSEAValid[,2])
n = sum(!is.na(RMSEAValid[,2]))
lines(RMSEA3Bvalid.ordered ,col='blue',(1:n)/n, type = 's')
abline(h = 0:1, v = 0.1,lty=2,col='black')

### GFI comparison ###
# Density plot #
densityGFI3Avalid <- density(GFIValid[,1])
densityGFI3Bvalid <- density(GFIValid[,2])
plot(densityGFI3Bvalid ,col='BLUE',ylim = c(0, 11), main ='GFI Histogram, Model 3A vs Model
3B',xlab='GFI value')
lines(densityGFI3Avalid   ,col='RED')
abline(v = 0.95,lty=2,col='black')

# CDF plot #
y <- seq(-10, 10, 0.01)
GFI3Avalid.ordered = sort(GFIValid[,1])
n = sum(!is.na(GFIValid[,1]))
plot(GFI3Avalid.ordered , (1:n)/n, ylim = c(0, 1),xlim = c(0.5, 1), col='red', type = 's', main = 'GFI CDFplots,
Model 3A vs Model 3B',xlab="GFI value")

GFI3Bvalid.ordered = sort(GFIValid[,2])
n = sum(!is.na(GFIValid[,2]))
lines(GFI3Bvalid.ordered ,col='blue',(1:n)/n, type = 's')
abline(h = 0:1,v = 0.95,lty=2,col='black')
```

```
par(mfrow=c(2,2))
### CFI comparison ###
# Density plot #
densityCFI3Avalid <- density(CFIValid[,1])
densityCFI3Bvalid <- density(CFIValid[,2])
plot(densityCFI3Bvalid,ylim = c(0, 3), col='BLUE', main ='CFI Histogram, Model 3A vs Model 3B',xlab='CFI
value')
lines(densityCFI3Avalid   ,col='RED')
abline(v = 0.95,lty=2,col='black')

# CDF plot #
y <- seq(-10, 10, 0.01)
CFI3Avalid.ordered = sort(CFIValid[,1])
n = sum(!is.na(CFIValid[,1]))
plot(CFI3Avalid.ordered , (1:n)/n, ylim = c(0, 1), col='red', type = 's', main = 'CFI CDFplots, Model 3A vs
Model 3B',xlab="CFI value")

CFI3Bvalid.ordered = sort(CFIValid[,2])
n = sum(!is.na(CFIValid[,2]))
lines(CFI3Bvalid.ordered ,col='blue',(1:n)/n, type = 's')
abline(h = 0:1,v = 0.95,lty=2,col='black')

### TLI comparison ###
# Density plot #
densityTLI3Avalid <- density(TLIValid[,1])
densityTLI3Bvalid <- density(TLIValid[,2])
plot(densityTLI3Bvalid,ylim = c(0, 1.6), col='BLUE', main ='TLI Histogram, Model 3A vs Model
3B',xlab='TLI value')
lines(densityTLI3Avalid   ,col='RED')
abline(v = 0.95,lty=2,col='black')

# CDF plot #
y <- seq(-10, 10, 0.01)
TLI3Avalid.ordered = sort(TLIValid[,1])
n = sum(!is.na(TLIValid[,1]))
plot(TLI3Avalid.ordered , (1:n)/n, ylim = c(0, 1), col='red', type = 's', main = 'TLI CDFplots, Model 3A vs
Model 3B',xlab="TLI value")

TLI3Bvalid.ordered = sort(TLIValid[,2])
n = sum(!is.na(TLIValid[,2]))
lines(TLI3Bvalid.ordered ,col='blue',(1:n)/n, type = 's')
abline(h = 0:1,v = 0.95,lty=2,col='black')
```

```
### AIC BIC plots ###
par(mfrow=c(2,2))
# AIC #
# Density plot #
densityAIC3A <- density(AICValid[,1])
densityAIC3B <- density(AICValid[,2])
plot(densityAIC3A ,col='red', ylim = c(0, 0.005),main ='AIC Histogram, Model 3A vs Model 3B',xlab='AIC
value')
lines(densityAIC3B ,col='blue')

# CDF plot #
y <- seq(-10, 10, 0.01)
AIC3Avalid.ordered = sort(AICValid[,1])
n = sum(!is.na(AICValid[,1]))
plot(AIC3Avalid.ordered , (1:n)/n, ylim = c(0, 1), col='red', type = 's', main = 'AIC CDFplots, Model 3A vs
Model 3B',xlab="AIC value")
AIC3Bvalid.ordered = sort(AICValid[,2])
n = sum(!is.na(AICValid[,2]))
lines(AIC3Bvalid.ordered ,col='blue',(1:n)/n, type = 's')

# BIC #
# Density plot #
densityBIC3Avalid <- density(BICValid[,1])
densityBIC3Bvalid <- density(BICValid[,2])
plot(densityBIC3Avalid ,ylim = c(0, 0.005),col='RED', main ='BIC Histogram, Model 3A vs Model
3B',xlab='BIC value')
lines(densityBIC3Bvalid ,col='BLUE')

# CDF plot #
y <- seq(-10, 10, 0.01)
BIC3Avalid.ordered = sort(BICValid[,1])
n = sum(!is.na(BIC3Avalid.ordered))
plot(BIC3Avalid.ordered , (1:n)/n, ylim = c(0, 1), col='red', type = 's', main = 'BIC CDFplots, Model 3A vs
Model 3B',xlab="BIC value")
BIC3Bvalid.ordered = sort(BICValid[,2])
n = sum(!is.na(BIC3Bvalid.ordered))
lines(BIC3Bvalid.ordered ,col='blue',(1:n)/n, type = 's')

###################### R codes for comparison tables and plots END###################
```