Make Some Noise: Methods for Generating Data From Imperfect Factor Models

A DISSERTATION
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Justin D. Kracht

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Niels G. Waller, Advisor

April 2022

Acknowledgements

I would like to acknowledge. . . Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas vel eros sed mauris porttitor semper nec a orci. Nullam vestibulum mi nec condimentum posuere. Pellentesque eget diam id sapien aliquet ullamcorper. Pellentesque blandit nec lectus ut mollis. Praesent in facilisis justo. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Sed eget congue leo, sed consequat libero. In rutrum malesuada nisi. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Morbi sollicitudin tortor ut sem facilisis mollis.

Dedication

This is for my mother who paved the way.

Abstract

This is an abstract that is the tldr; for my dissertation.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Covariance structure models (also called structural equation models) are widely used in psychological research (Cudeck, 1989). These models allow a structured covariance matrix to be represented as a matrix-valued function of a vector of parameters such that $\mathbf{\Omega} = \mathbf{\Omega}(\boldsymbol{\gamma})$, where $\mathbf{\Omega}$ is a $p \times p$ covariance matrix and $\boldsymbol{\gamma}$ is a vector of free model parameters. Stated another way, covariance structure models attempt to represent the structural connections between a set of unobserved latent variables (factors) and a set of observed variables that are indicators of the latent variables. When assessing model fit or estimating the dispersion of the estimated structural parameters, $\hat{\boldsymbol{\gamma}}$, researchers traditionally assume that the model holds perfectly in the population. It is assumed that there exists some vector $\boldsymbol{\gamma} = \boldsymbol{\gamma}_0$ such that the population covariance matrix $\mathbf{\Omega}$ can be perfectly reproduced (i.e., $\mathbf{\Omega} = \mathbf{\Omega}(\boldsymbol{\gamma}_0)$). If the model fits perfectly in the population, differences between the sample covariance matrix and the corresponding model-implied covariance matrix can only be due to sampling variability.

However, the assumption that the covariance structure is correctly specified in the population will nearly always be violated in practice (Browne et al., 2002; Cudeck & Henly, 1991; MacCallum et al., 2001; MacCallum & Tucker, 1991; Meehl & Waller, 2002; Tomarken & Waller, 2003). After all, "no mathematical model will fit real-world phenomena exactly" (MacCallum et al., 2001, p. 503) due, for example, to non-linearities in the relationships between factors and indicators, or to the effect of numerous minor factors of little theoretical interest (Cudeck & Browne, 1992). The idea that all models are imperfect representations of reality has been expressed many ways, perhaps most succinctly by Box's aphorism that "all models are wrong, but some are useful" (1987, p. 424). Further interesting discussions of the idea that models are only useful approximations of reality (and are often useful *because* they are approximations) can be found in both scientific and popular literature (Borges, 1998; Carroll, 1894; Eco, 1994; Gelman, 2008; Nester, 1996; Steele, 2008).

Recognizing that all models are literally false, and thus no covariance structure model will fit perfectly in the population, several authors have suggested that estimation error for these models can be attributed to two sources: error due to sampling variability and error due to model misfit (Browne & Cudeck, 1992; Cudeck & Henly, 1991; Tucker et al., 1969). This second type of error has been variously referred to as *model error* (MacCallum et al., 2001; Tucker et al., 1967, 1969), *error of approximation* (Cudeck & Henly, 1991), *specification error* (Satorra, 2015), or *adventitious error* (Wu & Browne, 2015a). Although some of these terms are related to specific views regarding the nature of model error (Tucker et al., 1969; c.f. Wu & Browne, 2015a), all of the terms refer to the discrepancy between the model-implied covariance matrix, $\mathbf{\Omega}$, and the error-perturbed population covariance matrix, $\mathbf{\Sigma} = \mathbf{\Omega} + \mathbf{E}$ (where $\mathbf{E}$ is a symmetric matrix representing the effects of model error). In this proposal, the term "model error" will generally be used to describe the discrepancy between the $\mathbf{\Sigma}$ and $\mathbf{\Omega}$.

Acknowledging model error is important because it can have significant implications for estimating covariance structure models. For instance, consider the traditional Chi-square test of exact model fit. Given a sample covariance matrix $\mathbf{S}$, the minimum objective function value for a hypothesized model $\hat{F} = F(\mathbf{S}, \mathbf{\Omega}(\hat{\gamma}))$ obtained by minimizing a discrepancy function $F(\mathbf{S}, \mathbf{\Omega}(\gamma))$ is assumed to follow a central Chi-square distribution when multiplied by $n = N - 1$, where N denotes the sample size (Olsson et al., 2004).[1] However, the test

[1]The maximum likelihood discrepancy function is commonly used, but other common discrepancy functions (e.g., generalized least squares, weighted least squares, asymptotically distribution free) will converge to the same minimum discrepancy values when the model is correctly specified and the observed variables

requires two stringent assumptions that are unlikely to both be satisfied in empirical settings: (a) that the observed variables are multivariate normal, and (b) that the model fits perfectly in the population (i.e., there is no model error; Browne, 1984). If (b) is not satisfied, then the test statistic $n\hat{F}$ will not follow a central Chi-square distribution and will lead to incorrect tests (Olsson et al., 2004). Moreover, Chi-square tests of exact fit are sensitive to sample size, with large sample sizes leading to almost certain rejection of the model, even with small amounts of misspecification (Bentler & Bonett, 1980; Tomarken & Waller, 2003; Yuan & Marshall, 2004). In any case, testing a model that is known not to be perfectly true against the null hypothesis of perfect fit would seem to have limited usefulness (Browne & Cudeck, 1992; Steiger, 2007).

Model error also has implications for covariance structure modeling beyond global tests of model fit. For instance, traditional methods of computing confidence intervals for model parameters assume that all error is sampling error (i.e., the model fits perfectly in the population). Thus, confidence intervals produced using these methods are overly-optimistic when model error is present (Wu & Browne, 2015a). Simulation studies have shown that the presence of model error can also impact parameter estimation for exploratory factor analysis (Briggs & MacCallum, 2003), dimensionality identification (Kracht & Waller, 2020), the behavior of confidence regions and fungible parameter estimates for structural equation models (Pek, 2012), and is important in other contexts as well (Beauducel & Hilger, 2016; de Winter & Dodou, 2016; Gnambs & Staufenbiel, 2016; Hsu et al., 2015; Trichtinger & Zhang, 2020).

These simulation studies represent a recent trend toward incorporating model error in Monte Carlo simulation studies involving covariance structure models. Including model error in these studies is important for at least two reasons. First, the addition of model error makes simulated data sets more representative of empirical data sets, which almost certainly do not have population covariance matrices that are perfectly fit by any simple covariance

are multivariate normal (Olsson et al., 2004).

structure model. Therefore, the inclusion of model error should lead to results that are more generalizable to empirical settings compared to Monte Carlo studies that do not include model error. Second, incorporating model error allows researchers to evaluate the robustness of methods when covariance structure models do not hold exactly.

Recognizing the importance of incorporating model error in Monte Carlo simulation studies, various authors have introduced methods for generating population covariance matrices with imperfect model fit. The three most popular of these methods were proposed by (a) Tucker, Koopman, and Linn (TKL; 1969), (b) Cudeck and Browne (CB; 1992), and (c) Wu and Browne (WB; 2015).

1.1 Model-Error Methods

1.1.1 The Tucker, Koopman, and Linn Method

One of the first model-error methods was developed by Tucker et al. (1969) in the context of common factor analysis. The common factor analysis model in terms of p manifest variables and k common factors can be written as

$$\mathbf{\Omega} = \mathbf{\Lambda}\mathbf{\Phi}\mathbf{\Lambda}' + \mathbf{\Psi}, \tag{1.1}$$

where $\mathbf{\Omega}$ is the model-implied $p \times p$ covariance matrix, $\mathbf{\Lambda}$ is the $p \times k$ factor-pattern matrix, $\mathbf{\Phi}$ is the $k \times k$ common factor covariance matrix, and $\mathbf{\Psi}$ is the $p \times p$ diagonal matrix containing the uniqueness variances. Without loss of generality, we can define all common factors as having means of zero and standard deviations of one so that $\mathbf{\Omega}$ has a unit diagonal (i.e., is a correlation matrix). Tucker, Koopman, and Linn (1969) proposed extending the common factor model to include many minor common factors of small effect in addition to the major common factors to represent "unsystematic or unknown aspects of the process that generates the data" (Cudeck & Browne, 1992, p. 358). The model they proposed can be written as

$$\mathbf{\Sigma} = \mathbf{\Lambda}\mathbf{\Phi}\mathbf{\Lambda}' + \mathbf{\Psi} + \mathbf{W}\mathbf{W}', \tag{1.2}$$

where $\mathbf{\Sigma}$ is the $p \times p$ error-perturbed population covariance matrix, $\mathbf{W}$ is the $p \times q$ matrix of minor common factor loadings, and all other terms are as previously defined. For convenience, we can define the observed variables to be in standard score form such that $\mathbf{\Sigma}$ has a unit diagonal. The combined influence of the q minor common factors represents the reliable common variance (and covariance) not accounted for by the major common factors and is considered to be due to model error.

The proportion of the uniqueness variance reapportioned to the minor common factors and how that variance is distributed to each minor common factor are determined by two user-specified parameters, $\nu_{\text{e}} \in [0, 1]$ and $\epsilon \in [0, 1]$, respectively. To create the matrix of minor factor loadings, $\mathbf{W}$, a $p \times q$ provisional matrix, $\mathbf{W}^*$, is first created such that the ith column of $\mathbf{W}^*$ consists of p independent samples from $\mathcal{N}(0, (1-\epsilon)^{i-1})$, where $\mathcal{N}(0, (1-\epsilon)^{i-1})$ denotes a normal distribution with a mean of zero and a standard deviation of $(1-\epsilon)^{i-1}$. Because the standard deviation of the ith column of $\mathbf{W}$ is given by $(1-\epsilon)^{i-1}$, values of ϵ close to zero result in columns with relatively equal variance, corresponding to approximately equipotent minor factors. On the other hand, values of ϵ close to one result in error variance primarily being distributed to the first minor factor, with the remaining variance distributed to the other minor factors in a decreasing geometric sequence.

To ensure that the minor common factors account for the specified proportion of uniqueness variance (ν_{e}), $\mathbf{W}^*$ is then scaled to create $\mathbf{W}$. This scaling is done in several steps. First, a diagonal matrix $\mathbf{\Psi}^*_{p \times p}$ is created such that

$$\mathbf{\Psi}^* = \mathbf{I}_p - \text{dg}(\mathbf{\Lambda}\mathbf{\Phi}\mathbf{\Lambda}'), \tag{1.3}$$

where $\text{dg}(\mathbf{\Lambda}\mathbf{\Phi}\mathbf{\Lambda}')$ is the diagonal matrix formed from the diagonal entries in $\mathbf{\Lambda}\mathbf{\Phi}\mathbf{\Lambda}'$ and $\mathbf{I}_p$ denotes a $p \times p$ identity matrix. Then the matrix $\mathbf{W}$ is formed using

$$\mathbf{W} = (\mathrm{dg}(\mathbf{W}^*\mathbf{W}^{*\prime})^{-1}\mathbf{\Psi}^*\nu_{\mathrm{e}})^{1/2}\mathbf{W}^*. \tag{1.4}$$

This process ensures that the q minor common factors account for the specified proportion of the variance not accounted for by the major common factors. The $\mathbf{W}$ matrix can then be used to create the diagonal matrix of unique variances, $\mathbf{\Psi} = \mathbf{I}_p - \mathrm{dg}(\mathbf{\Lambda\Phi\Lambda}' + \mathbf{WW}')$. The $\mathbf{\Lambda}$, $\mathbf{\Psi}$, and $\mathbf{W}$ matrices are then used to construct the population correlation matrix (with model error), $\mathbf{\Sigma}$, as shown in Equation ((1.2)). The TKL method is one of the most widely-used methods for generating covariance matrices with imperfect model fit (Beauducel & Hilger, 2016; Chung & Cai, 2019; de Winter & Dodou, 2016; Gnambs & Staufenbiel, 2016; Kracht & Waller, 2020; Lorenzo-Seva & Ferrando, 2020; Lorenzo-Seva & Ginkel, 2016; Myers et al., 2015).[2]

Although the original TKL method is still most commonly used in simulation studies, at least two variants of the original TKL method have since been developed. First, Hong (1999) introduced a variation of the TKL method that allowed for minor common factors that were correlated with each other and with the major common factors. In Hong's model, the population covariance matrix can be written as

$$\mathbf{\Sigma} = \mathbf{LBL}' + \mathbf{\Psi}, \tag{1.5}$$

where $\mathbf{L} = \begin{bmatrix}\mathbf{\Lambda} & \mathbf{W}\end{bmatrix}$ is the super matrix containing the major and minor factor loadings, and $\mathbf{\Psi}$ is the diagonal matrix of uniqueness variances, as defined in Equation ((1.1)). The matrix $\mathbf{B}$ is the correlation matrix for the major and minor factors such that

$$\mathbf{B} = \begin{bmatrix}\mathbf{\Phi} & \mathbf{\Upsilon} \\ \mathbf{\Upsilon}' & \mathbf{\Gamma}\end{bmatrix}, \tag{1.6}$$

[2]Tucker, Koopman, and Linn (1969) was cited 163 times as of March 11, 2021, according to citation counts provided by Web of Science and Crossref.

where $\mathbf{\Phi}$ is the $k \times k$ major factor correlation matrix, $\mathbf{\Upsilon}$ is the $k \times q$ matrix of correlations between the major and minor factors, and $\mathbf{\Gamma}$ is the $q \times q$ minor factor correlation matrix.

Hong's (1999) article has been cited a number of times since its publication[3], but only one published study (Porritt, 2015) has applied Hong's method. In Porrit's (2015) simulation study, all major and minor factor correlations were fixed at .3, following the example given by Hong (1999). However, neither Porrit (2015) nor Hong (1999) directly compared Hong's method with the original TKL method. Therefore, it remains unclear whether modeling major-minor factor correlations is important when simulating covariance matrices with model error.

A second variant of the TKL method was recently introduced by Trichtinger & Zhang (2020). Specifically, Trichtinger and Zhang's method adapted the TKL method to allow the simulation of covariance matrices with model error for multivariate time series data. Trichtinger and Zhang (2020) first introduced a novel test statistic appropriate for P-technique factor analysis. They then used their adapted TKL method to generate data with model error in a Monte Carlo simulation study to evaluate the empirical characteristics of their test statistic. Although their adapted TKL method has not yet been used in any other published simulation studies, it demonstrates that it is possible to extend the TKL method to other types of models.

1.1.2 The Cudeck and Browne (CB; 1992) Method

An alternative to the TKL method for modeling imperfect model fit in simulation studies was described by Cudeck and Browne (1992). These authors agreed with Tucker, Koopman, and Linn (1969) that no simple factor analysis model is likely to fit exactly in the population and that Monte Carlo simulation studies conducted to evaluate statistical methods should incorporate model error to test robustness against imperfect model fit. However, Cudeck and Browne wanted a model-error method that was more flexible than the TKL method

[3]Hong (1999) was cited twelve times as of March 11, 2021, according to Web of Science.

and that allowed the user to specify the desired discrepancy function value to control the amount of model error. Therefore, Cudeck and Browne (1992) proposed a new method of generating data from models with imperfect fit, building on the work of Tucker, Koopman, and Linn (1969). They developed their new approach to satisfy three desiderata: First, that the approach is general to covariance structure models and not only factor analysis models; Second, that the approach allows the pre-specification of the amount of model error in terms of the maximum likelihood or least squares discrepancy function value, $F(\boldsymbol{\Sigma}, \boldsymbol{\Omega}(\boldsymbol{\gamma})) = \delta$; Third, that the minimizer of the discrepancy function is a specified vector of model parameters, $\boldsymbol{\gamma} = \boldsymbol{\gamma_0}$.

The Cudeck-Browne method (referred to as CB hereafter) works as follows. A $p \times p$ population covariance matrix is defined as the sum

$$\boldsymbol{\Sigma} = \boldsymbol{\Omega}(\boldsymbol{\gamma}) + \mathbf{E}, \tag{1.7}$$

where $\boldsymbol{\Omega}(\boldsymbol{\gamma})$ is a matrix-valued function of a vector of parameters, $\boldsymbol{\gamma}$, and $\mathbf{E}$ is a $p \times p$ symmetric matrix such that Equation ((1.7)) is positive definite. Moreover, let $\boldsymbol{\Sigma}_0 = \boldsymbol{\Omega}(\boldsymbol{\gamma}_0) + \mathbf{E}$ be the population covariance matrix for a particular vector of model parameters, $\boldsymbol{\gamma}_0$. The CB method works by finding an $\mathbf{E}$ matrix such that the discrepancy function $F(\boldsymbol{\Sigma}_0, \boldsymbol{\Omega}(\boldsymbol{\gamma}_0))$ is minimized at $\boldsymbol{\gamma}_0$ and the minimum is equal to a pre-specified value, δ. Cudeck and Browne (1992) considered discrepancy functions of the form

$$F(\boldsymbol{\Sigma}, \boldsymbol{\Omega}(\boldsymbol{\gamma})) = \frac{1}{2}\operatorname{tr}[\mathbf{Z}^{-1}(\boldsymbol{\Sigma} - \boldsymbol{\Omega}(\boldsymbol{\gamma}))^2], \tag{1.8}$$

where the fixed matrix $\mathbf{Z}$ does not depend on $\mathbf{E}$. Note that when $\mathbf{Z} = \mathbf{I}_p$, Equation ((1.8)) is the discrepancy function for ordinary least squares. The discrepancy function for ordinary-theory maximum likelihood can be written as

$$F_{\mathrm{ML}}(\boldsymbol{\Sigma}, \boldsymbol{\Omega}(\boldsymbol{\gamma})) = \ln|\boldsymbol{\Omega}(\boldsymbol{\gamma})| - \ln|\boldsymbol{\Sigma}| + \operatorname{tr}[\boldsymbol{\Sigma}\boldsymbol{\Omega}(\boldsymbol{\gamma})^{-1}] - p, \tag{1.9}$$

where $|\boldsymbol{\Sigma}|$ is the determinant of $\boldsymbol{\Sigma}$. Cudeck and Browne (1992) showed that the minimizer of Equation ((1.8)) is the same as the minimizer of Equation ((1.9)) when $\mathbf{Z} = \boldsymbol{\Omega}(\boldsymbol{\gamma}_{\text{ML}})$, where $\boldsymbol{\gamma}_{\text{ML}}$ is the minimizer of the maximum likelihood discrepancy function in Equation ((1.9)). Thus, Equation ((1.8)) is a general form of the discrepancy function that is equivalent to either the least squares discrepancy function or the maximum likelihood discrepancy function, depending on the value of $\mathbf{Z}$.

Recall that one objective of the CB method is to ensure that the discrepancy function $F(\boldsymbol{\Sigma}_0, \boldsymbol{\Omega}(\boldsymbol{\gamma}))$ is minimized when $\boldsymbol{\gamma} = \boldsymbol{\gamma}_0$. To do this, it is necessary to find an $\mathbf{E}$ matrix such that the gradient $\partial F(\boldsymbol{\Sigma}_0, \boldsymbol{\Omega}(\boldsymbol{\gamma}))/\partial \boldsymbol{\gamma} = \mathbf{0}$. Cudeck and Browne (1992) showed the gradient can be written as

$$\frac{\partial F(\boldsymbol{\Sigma}_0, \boldsymbol{\Omega}(\boldsymbol{\gamma}))}{\partial \boldsymbol{\gamma}} = \mathbf{B}'\tilde{\mathbf{e}} \tag{1.10}$$

where $\mathbf{B}$ is a $p \times p$ symmetric matrix that depends on both $\mathbf{Z}$ and $\dot{\boldsymbol{\Sigma}}_i = [\partial \boldsymbol{\Sigma}(\boldsymbol{\gamma})/\partial \gamma_i]$, and $\tilde{\mathbf{e}} = \text{vecs}\,[\boldsymbol{\Sigma}_0 - \boldsymbol{\Omega}(\boldsymbol{\gamma})]$. The vecs operator is defined such that for a symmetric matrix, $\mathbf{A}$, $\text{vecs}\,\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{22} & a_{13} & \dots & a_{pp} \end{bmatrix}'$. Put another way, the vecs $\mathbf{A}$ operator returns a vector of the stacked upper-diagonal elements of $\mathbf{A}$ (including the diagonal). Written in this form, the gradient can be set equal to the null vector, $\mathbf{B}'\tilde{\mathbf{e}} = \mathbf{0}|_{\boldsymbol{\gamma}=\boldsymbol{\gamma_0}}$ and solved for $\tilde{\mathbf{e}} = \text{vecs}\,\tilde{\mathbf{E}}$. To find a suitable $\tilde{\mathbf{e}}$, let $\mathbf{y}$ be a non-null $\frac{1}{2}(p^2 + p) \times 1$ vector. Then the difference $\tilde{\mathbf{e}} = \mathbf{y} - \mathbf{B}(\mathbf{B}'\mathbf{B})^{-1}\mathbf{B}'\mathbf{y}$ gives an $\tilde{\mathbf{e}}$ vector such that $\mathbf{B}'\tilde{\mathbf{e}} = \mathbf{0}|_{\boldsymbol{\gamma}=\boldsymbol{\gamma_0}}$.

After finding $\tilde{\mathbf{E}}$ such that $F(\boldsymbol{\Sigma}_0, \boldsymbol{\Omega}(\boldsymbol{\gamma}))$ is minimized at $\boldsymbol{\gamma} = \boldsymbol{\gamma}_0$, the next step is to ensure that the minimum is equal to a specified value, δ. The population root mean square error of approximation (RMSEA; Steiger, 1990) value is related to the objective function value by $\varepsilon = \sqrt{F(\boldsymbol{\Sigma}_0, \boldsymbol{\Omega}(\boldsymbol{\gamma}))/df}$, where ε and df denote the RMSEA and model degrees of freedom, respectively. Therefore, δ is generally selected to produced a desired RMSEA value such that $\delta = \varepsilon^2 df$. Cudeck and Browne defined the error matrix as $\mathbf{E} = \kappa\tilde{\mathbf{E}}$, where κ is a scaling term that is chosen so that the value of the objective function at its minimum is equal to δ. When

$\kappa = 0$, $\mathbf{\Sigma} = \mathbf{\Omega}$, whereas larger values of κ lead to larger discrepancy function values. Cudeck and Browne (1992) furthermore proved that $\boldsymbol{\gamma}_0$ is the global minimizer of $F(\mathbf{\Sigma}_0, \mathbf{\Omega}(\boldsymbol{\gamma}))$ as long as κ is not too large (pp. 360–361). Thus, their second desideratum (i.e., that $\boldsymbol{\gamma}_0$ be the minimizer of the objective function) and third desideratum (i.e., that the value of the discrepancy function at its minimum is δ) are both satisfied.

The CB method is appealing for use in simulation studies for a number of reasons. First, it allows the user to specify a desired RMSEA value. Second, unlike the TKL method, the CB method does not have any tuning parameters that need to be chosen (other than the target RMSEA). Third, the CB method is easily extendable to many types of covariance structure models. Likely because of these advantages, the CB method has been used in a number of Monte Carlo simulation studies (Lai, 2020a, 2018, 2020b, 2020c, 2019, 2020d; Lai & Zhang, 2017; Montoya & Edwards, 2020; Xia, 2021).

Although it is appealing for simulation work, the CB method carries with it an assumption about the nature of model error that might or might not be considered reasonable for empirical data sets. Recall that a requirement of the CB method is that the discrepancy function $F(\mathbf{\Sigma}_0, \mathbf{\Omega}(\boldsymbol{\gamma}_0))$ is minimized when $\boldsymbol{\gamma} = \boldsymbol{\gamma}_0$. When a model is misspecified, there does not exist any $\boldsymbol{\gamma}_0$ such that $\mathbf{\Sigma}_0 = \mathbf{\Omega}(\boldsymbol{\gamma}_0)$. However, the maximum likelihood parameter estimate $\hat{\boldsymbol{\gamma}}_{\text{ML}}$ is still consistent toward the minimizer of the maximum likelihood discrepancy function under mild regularity conditions (Shapiro, 1983, Theorem 5.4; 2007, sec. 5.3; Wu & Browne, 2015a). Moreover, if a sample covariance matrix $\mathbf{S}$ is an unbiased estimator of $\mathbf{\Sigma}_0$, then the expected value of $\mathbf{S}$ is $\mathbf{\Sigma}_0$. Taken together, these properties indicate that maximum likelihood parameter estimates converges to $\boldsymbol{\gamma}_0$ as $N \to \infty$ under the CB framework.

This result reflects the view that the "true" population parameter values are simply the parameter values that are obtained by fitting a model to $\mathbf{\Sigma}$ using a particular discrepancy function. In this view, the discrepancy between the $\mathbf{\Sigma}$ and the model-implied covariance matrix obtained from analyzing $\mathbf{\Sigma}$ ($\hat{\mathbf{\Sigma}}$) is of primary interest, not the discrepancy between $\mathbf{\Sigma}$ and the implied covariance matrix for some ideal model ($\mathbf{\Omega}$). Because the CB method ensures that $\boldsymbol{\gamma}_0$ is a minimizer of the discrepancy function, $\hat{\boldsymbol{\Sigma}} = \boldsymbol{\Omega}$. Thus, it makes sense that the population parameters ($\boldsymbol{\gamma}_0$) will be perfectly recovered as $N \to \infty$.

1.1.3 The Wu and Browne (WB; 2015) Method

A third model-error method was introduced by Wu and Browne (2015) and is unique among the three approaches because it represents model error as a random effect rather than as a fixed quantity. Moreover, the TKL and CB model-error methods were developed for use in simulation studies. In contrast, the WB method was motivated by Wu and Browne's development of a method for estimating confidence intervals for model parameters, taking into account variability due to model error.

Before describing Wu and Browne's (2015) estimation method, it will be useful to define two important terms. The term *model discrepancy*, as used by Wu and Browne (2015), corresponds to what has been previously referred to as model error (i.e., the difference between the error-perturbed and model-implied population covariance matrices). The term *adventitious error* is used to describe the process underlying model discrepancy (Wu & Browne, 2015a). In Wu and Browne's (2015) conceptualization, model discrepancy arises from differences between two populations: an operational population from which the observed sample is representative, and a theoretical general population. The theory (as represented by the covariance structure model) is hypothesized to hold exactly in the theoretical general population, but not in the operational population (Wu & Browne, 2015a, 2015b). The general population might also be referred to as the *ideal* population. This is not the terminology used by Wu and Browne (2015), but more clearly reflects their view of the general population as being "...contained in the Platonic Aether" (attributed to Michael C. Edwards in Wu & Browne, 2015b, p. 620) as opposed to "a mundane collection of people that can be reached through more complicated designs" (Wu & Browne, 2015b, p. 621).

Wu and Browne (2015) argued that the discrepancy between the operational and ideal population models is a source of variation that is not reflected in traditional methods for fitting covariance structures to sample covariance matrices. These approaches generally fit a covariance structure $\boldsymbol{\Omega}(\boldsymbol{\gamma})$ to a sample covariance matrix $\mathbf{S}$ by minimizing a discrepancy function $F(\mathbf{S}, \boldsymbol{\Omega}(\boldsymbol{\gamma}))$. For instance, a common choice of discrepancy function is the maximum likelihood discrepancy function,

$$F_{\mathrm{ML}}(\mathbf{S}, \boldsymbol{\Omega}(\boldsymbol{\gamma})) = \ln|\boldsymbol{\Omega}(\boldsymbol{\gamma})| - \ln|\mathbf{S}| + \mathrm{tr}[\mathbf{S}\boldsymbol{\Omega}(\boldsymbol{\gamma})^{-1}] - p, \tag{1.11}$$

Note that Equation ((1.11)) is equivalent to Equation ((1.9)) with $\mathbf{S}$ substituted for $\boldsymbol{\Sigma}$. The discrepancy function is called "maximum likelihood" because minimizing the function is equivalent to maximizing the likelihood function for the Wishart distribution, $\mathrm{W}_p(\boldsymbol{\Omega}(\boldsymbol{\gamma})/n, n)$. This Wishart distribution is the sampling distribution of $\mathbf{S}$ under the assumption that $\boldsymbol{\Sigma} = \boldsymbol{\Omega}(\boldsymbol{\gamma})$ for some $\boldsymbol{\gamma} = \boldsymbol{\gamma}_0$, and the assumption of normality (Wu & Browne, 2015a). The asymptotic distribution of the maximum likelihood parameter estimate, $\hat{\boldsymbol{\gamma}}_{\mathrm{ML}}$ can then be derived (e.g., Shapiro, 2007, Theorem 5.5, p. 249), and confidence intervals for parameters can be constructed (Jöreskog, 1969).

Unfortunately, using maximum likelihood estimation requires assumptions that are unlikely to hold in many applied settings. First, the maximum likelihood discrepancy function is derived using normal distribution theory and can be sensitive to violations of normality (Browne & Shapiro, 1988). However, maximum likelihood estimates can also be derived using a model that does not require any distributional assumptions (Howe, 1955; Mulaik, 2009, pp. 214–215) and normal theory methods have been shown to be robust under certain circumstances (Browne & Shapiro, 1988). A more troublesome issue highlighted by Wu and Browne (2015) is that using the maximum likelihood discrepancy function defined in Equation ((1.11)) implicitly assumes that the theorized model holds perfectly in the population and that all differences between the sample and population covariance matrices are attributable to sampling error (Briggs & MacCallum, 2003; Wu & Browne, 2015a). Because adventitious error is not considered as a source of random variation, the variability estimates (and therefore confidence intervals) of parameter estimates and test statistics will be underestimated when using the traditional approach to model estimation (Wu & Browne, 2015a).

Unlike the traditional approach, Wu and Browne's (2015) estimation method accounts for variability due to adventitious error by modeling adventitious error as a random effect with a distribution. The estimated dispersion parameter of this distribution can then be used as a measure of model misspecification. The statistical basis for their model is as follows. First, under the assumption of normality, the sample covariance matrix $\mathbf{S}$ has a Wishart distribution such that

$$(\mathbf{S}|\mathbf{\Sigma}) \sim \mathrm{W}_p(\mathbf{\Sigma}/n, n), \tag{1.12}$$

where n is the degrees of freedom. As opposed to the traditional method where $\mathbf{S}$ is assumed to be an unbiased estimator of the model-implied population covariance matrix $\mathbf{\Omega}(\boldsymbol{\gamma})$, here $\mathbf{S}$ is instead assumed to be an unbiased estimator of the error-perturbed population covariance matrix, $\mathbf{\Sigma}$. $\mathbf{\Sigma}$ is then assumed to follow an inverse-Wishart distribution such that

$$(\mathbf{\Sigma}|\mathbf{\Omega}, m) \sim \mathrm{W}_p^{-1}(m\mathbf{\Omega}, m), \tag{1.13}$$

where m is a continuous precision parameter such that $m > p-1$ (Wu & Browne, 2015b). Wu and Browne (Wu & Browne, 2015a) also introduced the inverse of this precision parameter, $v = 1/m \in [0, (p-1)^{-1})$, which gives the dispersion of the adventitious error and can also be interpreted as a measure of misspecification. In particular, Wu and Brown (2015a, p. 580) show that $v \approx \varepsilon^2$, where ε denotes the RMSEA. Because v has an upper bound of $1/(p-1)$, Wu and Browne suggested using $\sqrt{\tilde{v}} = (m - p + 1)^{-1/2}$ as the criterion of model admissibility. Specifically, they stated that $\sqrt{\tilde{v}} = 0.05$ is indicative of good model fit, values of $\sqrt{\tilde{v}}$ between 0.05 and 0.08 are indicative of acceptable model fit, and values above 0.08 are indicative of unacceptable model fit.

Wu and Browne's (2015a) model therefore has two parameters, $\boldsymbol{\gamma}$ and v, that require estimation. They showed that these parameters can be estimated by maximizing the likelihood function corresponding to the marginal distribution of the sample covariance matrix, $(\mathbf{S}|\boldsymbol{\Omega}, m)$, with the population covariance matrix $\boldsymbol{\Sigma}$ integrated out. Wu and Browne's choice of a conjugate distribution for $\boldsymbol{\Sigma}$ means that the marginal distribution of the sample covariance matrix follows a Type II matrix-variate beta distribution (Gupta & Nagar, 2000, Chapter 5),

$$(\mathbf{S}|\boldsymbol{\Omega}, m) \sim \mathbf{B}_p^{\mathrm{II}}\left(\frac{n}{2}, \frac{m}{2}, \frac{m}{n}\boldsymbol{\Omega}\right), \tag{1.14}$$

from which the probability density function and log-likelihood functions can be obtained (see Wu & Browne, 2015a for additional details). The parameter estimate obtained by maximizing the beta marginal likelihood (or equivalently, minimizing the negative twice beta log-likelihood) is referred to as the maximum beta likelihood estimate (MBLE) by Wu and Browne (2015a).

In addition to showing how to estimate $\boldsymbol{\gamma}$ and v, Wu and Browne (2015) derived sampling distributions and confidence intervals for $\hat{\boldsymbol{\gamma}}$ and $\hat{v}$. An important advantage of Wu and Browne's method over more traditional estimation methods is that their confidence intervals for $\hat{\boldsymbol{\gamma}}$ account for the amount of model error indicated by $\hat{v}$. They provided simulation evidence showing that confidence interval coverage rates for both estimated parameters were close to their nominal values, at least when n and m were relatively large and when $(\boldsymbol{\Sigma}|\boldsymbol{\Omega}, m) \sim \mathrm{W}_p^{-1}(m\boldsymbol{\Omega}, m)$. Moreover, they also showed that coverage rates for 95% confidence intervals estimated using their method were much closer to the nominal levels compared to confidence intervals estimated using the traditional approach.

Thus far, this section has focused on Wu and Browne's (2015) method for estimating parameters and constructing confidence intervals for covariance structure models. However, as noted previously, Wu and Browne's model assumes that a covariance matrix with imperfect model fit is a random sample from an inverse-Wishart distribution, as shown in Equation ((1.13)). Given a model-implied population covariance matrix $\mathbf{\Sigma}$ and some chosen value of the precision parameter m, covariance matrices with imperfect fit can be easily sampled from an appropriate inverse-Wishart distribution using statistical software such as R (R Core Team, 2021), MATLAB (The Mathworks, 2019), or Julia (Bezanson et al., 2017). The degree of model error (in terms of RMSEA) can be controlled to some extent by choosing an appropriate value of m, which Wu and Browne showed has a relationship with RMSEA such that $\sqrt{1/m} = \sqrt{v} \approx \varepsilon$ (Wu & Browne, 2015a, Eq. 16, p. 580).

1.2 Population Model Fit Indices

In addition to being able to generate error-perturbed covariance matrices, it is important to be able to quantify the lack-of-fit between an error-perturbed population covariance matrix and a model-implied covariance matrix. Before describing the various model-fit indices that have been used to quantify model error, I must first describe two different perspectives of model fit as it relates to model error. In the first, model error is quantified as the lack-of-fit between $\mathbf{\Sigma}$ and $\mathbf{\Omega}$. In the second, model error is quantified as the lack-of-fit between $\mathbf{\Sigma}$ and $\hat{\mathbf{\Omega}}$, the implied covariance matrix from fitting a particular model to $\mathbf{\Sigma}$. These two perspectives are equivalent in the special case when $\hat{\mathbf{\Omega}} = \mathbf{\Omega}$, which occurs when the CB method is used (because $\boldsymbol{\gamma}_0$ is required to be a minimizer of the chosen discrepancy function). For the purposes of this dissertation, I will focus primarily on the first model fit perspective. However, all of the population model fit indices described in this section can be used to reflect either perspective.

As described in the previous section, the population RMSEA value is often used to describe the lack-of-fit between population covariance matrices due to model error. Other fit indices such as the Tucker-Lewis Index (TLI; Tucker & Lewis, 1973), the Comparative Fit Index (CFI; Bentler, 1990), and the Standardized Root Mean Square Residual (SRMR;

Hu & Bentler, 1999) have also be used for this purpose. Although far from an exhaustive list, these fit indices are among the most popular in the psychometric literature and can be divided into two general categories (Hu & Bentler, 1999). First, RMSEA and SRMR are absolute fit indices that report model fit in terms of the difference between a particular covariance matrix and a model-implied covariance matrix, without using a reference or baseline model for comparison. In the population context, absolute fit indices answer the question, "How different is $\mathbf{\Sigma}$ to $\mathbf{\Omega}$?" On the other hand, the TLI and CFI are incremental fit indices, which reflect the improvement of fit of a particular model compared to a (restricted, nested) baseline model. The null or independence model is often used as the baseline model, answering the question, "How well is my model doing, compared with the worst model that there is?" (Miles & Shevlin, 2007, p. 870).

RMSEA, CFI, TLI, and SRMR are generally defined in terms of the sample covariance matrix, but can be expressed in their population form (i.e., with reference to $\mathbf{\Sigma}$) as follows. Let F_h and F_b be the minimized discrepancy function values for the hypothesized and baseline models (respectively) at the population level. Furthermore, let df_h and df_b be the degrees of freedom for the hypothesized and baseline models. Then the population RMSEA value is given by

$$\text{RMSEA} = \varepsilon = \sqrt{\frac{F_h}{df_h}}, \tag{1.15}$$

the population CFI value is given by

$$\text{CFI} = 1 - \frac{F_h}{F_b}, \tag{1.16}$$

and the population TLI value is given by

$$\text{TLI} = 1 - \frac{F_h/df_h}{F_b/df_b}. \tag{1.17}$$

Unlike RMSEA, CFI, and TLI, the population SRMR does not depend on discrepancy function values and is given by

$$\text{SRMR} = \sqrt{\frac{1}{p(p+1)/2} \sum_{i \leq j} \left(\frac{\sigma_{ij} - \omega_{ij}}{\sqrt{\sigma_{ii}\sigma_{jj}}} \right)^2}, \tag{1.18}$$

where p is the number of observed variables, and σ_{ij} and ω_{ij} are the i, jth elements of $\mathbf{\Sigma}$ and $\mathbf{\Omega}$, respectively (Maydeu-Olivares, 2017; Pavlov et al., 2021; Xia & Yang, 2019). Note that when $\mathbf{\Sigma}$ and $\mathbf{\Omega}$ are constrained to be correlation matrices with unit diagonals, the correlation root mean residual (CRMR; Kenneth A. Bollen, 1989; Ogasawara, 2001) is appropriate:

$$\text{CRMR} = \sqrt{\frac{1}{p(p-1)/2} \sum_{i<j} (\sigma_{ij} - \omega_{ij})^2}. \tag{1.19}$$

Although all of the fit indices discussed here have been used to quantify model error in simulation studies (e.g., Lai, 2020b, 2020c; Lai & Zhang, 2017; Xia et al., 2016), the population RMSEA is used most often. Indeed, many simulation studies using the TKL, CB, or WB model-error methods have grouped simulated covariance matrices into model fit categories based on highly-cited "rule-of-thumb" RMSEA cutoff values. For instance, Lorenzo-Seva and Ferrando (2020) used RMSEA values of 0.065 to represent models with fair fit, citing Browne and Cudeck (1992). Myers et al. (2015) used RMSEA values of 0.025, 0.065, and 0.090 to categorize models as having very good fit, fair fit, and poor fit (respectively), citing Browne and Cudeck (1992), Steiger (1989), and Jackson (2009). According to Lai and Green (2016, p. 220):

> The most widely used cutoffs for RMSEA yield the following interpretations: (a) Values less than .05 (Browne & Cudeck, 1992) or .06 (Hu & Bentler, 1999) suggest 'good' fit; (b) values between .05 and .10 suggest 'acceptable' fit (Browne & Cudeck, 1992; MacCallum, Browne, and Sugawara, 1996); and (c) values larger than .10 suggest 'bad' fit (Browne & Cudeck, 1992).

However, rule-of-thumb cutoffs based on other common fit indices such as CFI can lead to conclusions about model fit that disagree with those indicated by RMSEA.

The problem of disagreement among fit indices was explored in detail by Lai and Green (2016), who derived necessary and sufficient conditions for disagreement between CFI and RMSEA (when used with common cutoff values). For instance, Lai and Green (2016) showed that "good" RMSEA values and "bad" CFI values (i.e., RMSEA ≤ 0.05 and CFI ≤ 0.9) will occur when $df_h \leq 400F_b \leq 10df_h$, where df_h denotes the model degrees of freedom and F_b is the discrepancy function value for the baseline null (independence) model. Lai and Green (2016) also showed that $F_b = -\ln|\mathbf{S}|$ for an observed correlation matrix and that $\ln|\mathbf{S}|$ will be large (and $-\ln|\mathbf{S}|$ small) when the elements of $\mathbf{S}$ are close to zero. Thus, models will have "good" RMSEA but "bad" CFI when the elements of $\mathbf{S}$ are small (but large enough that $F_b > df_h/400$) and when the model degrees of freedom value is large (but small enough that $df_h < 400F_b$; Lai and Green, 2016). Lai and Green (2016) concluded that disagreement between CFI and RMSEA does not reflect a failure on the part of either fit index. Rather, disagreement occurs because the two fit indices evaluate model fit from different perspectives and the cutoffs used to make qualitative judgments about model fit are arbitrary.

Unfortunately, disagreement between fit indices is a problem for researchers attempting to simulate covariance matrices with some particular qualitative level of model fit. Consider, for instance, a simulated covariance matrix with RMSEA $= 0.04$ and CFI $= 0.78$.[4] An RMSEA value of 0.04 indicates good model fit based on most rule-of-thumb cutoff values (Browne & Cudeck, 1992). However, CFI values less than 0.9 are seldom considered to indicate acceptable fit (Lai & Green, 2016). How should this covariance matrix (and others like it) be categorized in a simulation study? No straightforward answer is apparent. The essential problem, as stated by Kline (2011), is that "*there is no such thing as a magical, single-number summary that says everything worth knowing about model fit*" (p. 193).

[4]This is not just a hypothetical example. Simulated correlation matrices leading to similar RMSEA and CFI values were encountered in simulations conducted by Kracht and Waller (2020).

Therefore, a common recommendation is to report multiple fit indices to get a more complete picture of model fit (Kline, 2011; Lai & Green, 2016)

Despite the recommendation to report multiple fit indices, only a handful of simulation studies that incorporate error-perturbed covariance matrices report fit indices other than RMSEA (Kracht & Waller, 2020; Lai, 2020a, 2018, 2020b, 2019; Lai & Green, 2016; Lai & Zhang, 2017). The reliance on a single model fit index (usually RMSEA) when generating error-perturbed covariance matrices can be explained by the characteristics of the TKL, CB, and WB model-error methods.

In the case of the TKL method, little is known about what values of the TKL parameters (ϵ and ν_{e}) are reasonable for generating data that are representative of empirical data. Without clear guidance about which parameter values to use, researchers using the TKL method in simulation studies have often relied on RMSEA values to determine whether resulting correlation matrices were representative of empirical data sets. For instance, Briggs & MacCallum (2003) used the TKL method to generate data sets they categorized as having either good or moderate model fit. In their simulation study, error-perturbed covariance matrices were retained or rejected based on their RMSEA values. A matrix was retained if the RMSEA resulting from a maximum likelihood factor analysis fell within the range .03–.049 for the good fit condition and within the range .07–.089 for the moderate model fit condition.

Other researchers have also used RMSEA as a measure of model misfit when using the TKL method in Monte Carlo simulation studies (Gnambs & Staufenbiel, 2016; Kracht & Waller, 2020; Lorenzo-Seva & Ferrando, 2020; Myers et al., 2015; Preacher et al., 2013; Preacher & MacCallum, 2002). One difficulty with this approach is that it can be challenging to choose values of ϵ and ν_{e} that lead to desired RMSEA values while varying factor model characteristics (e.g., number of factors, factor salience, factor correlations, etc.). Manually choosing parameters that routinely lead to desired RMSEA values and desired values of another fit statistic (such as CFI) is even more challenging. Thus, researchers who use the

TKL method in simulation studies generally select the values of ϵ and ν_{e} based on only RMSEA.

Researchers who choose to instead generate error-perturbed covariance matrices using the CB method often use RMSEA as a measure of model misfit because the CB method allows them specify a desired discrepancy function value (and therefore a desired RMSEA value). However, the CB method has no other user-specified parameters than can be changed to influence model fit indices that measure other aspects of model fit (such as CFI). Lai and Green (2016) used the CB method to generate covariance matrices with specified RMSEA and CFI values, but did so by updating the factor loadings and uniqueness variances in the population model while holding RMSEA constant. Although this approach is effective, it has the significant drawback of not allowing researchers to systematically vary model parameters across experimental conditions.

Similar to the CB method, the WB method allows the user to choose a desired RMSEA value but does not provide an easy way to specify a desired a CFI value (or any alternative fit index). The WB method is also less precise than the CB method in the sense that it only allows the user to specify a distribution for the model-perturbed covariance matrices and thus does not guarantee that the RMSEA value for a simulated covariance matrix will be very close to the target value.

Chapter 2

A Method for Generating Error-Perturbed Covariance Matrices with Specified Levels of Model Fit

Given that model fit cannot be fully described by a single fit index, it is desirable to have a model fit method that would allow for the simultaneous specification of multiple fit indices (e.g., RMSEA and CFI). To date, RMSEA has been used almost exclusively to quantify the amount of model error introduced by a model-error method, and is the only fit index that can be specified in advance when using the CB and WB methods (Briggs & MacCallum, 2003; Cudeck & Browne, 1992; Wu & Browne, 2015a).

In this section, I propose a procedure based on the TKL method that uses optimization to find parameter values that lead to error-perturbed covariance matrices with RMSEA and CFI values that are close to user-specified target values. RMSEA and CFI are used because they are among the most common model fit indices and reflect different aspects of model fit (absolute and incremental model fit). However, in theory the procedure could be extended to work with any two fit indices.[1] The proposed procedure also allows the user to impose constraints on the number of large minor factor loadings to ensure a clear delineation between major and minor factors in the TKL framework. Because the optimization procedure allows

[1]In fact, the procedure could easily be extended to include three or more fit indices. However, the marginal benefit of including additional fit indices is unlikely to be large and could make optimization more difficult.

for RMSEA and CFI targets and constraints on factor loadings, I will refer to it as the "multiple-target optimization method" or "the multiple-target method."

The proposed procedure uses the limited memory Broyden-Fletcher-Goldfarb-Shanno optimization algorithm with box constraints (L-BFGS-B; Byrd et al., 1995) to minimize the objective function

$$G(\nu_{\mathrm{e}}, \epsilon) = b_1 \frac{[\varepsilon - \varepsilon_T]^2}{\varepsilon_T^2} + b_2 \frac{[\mathrm{CFI} - \mathrm{CFI}_T]^2}{(1 - \mathrm{CFI}_T)^2} + \mathbf{1}_{\boldsymbol{W}} \lambda. \tag{2.1}$$

where $0 \leq \nu_{\mathrm{e}} \leq 1$, $0 \leq \epsilon \leq 1$, b_1 and b_2 are user-specified weights constrained to sum to one. Setting $b_1 = b_2 = 0.5$ places equal weight on RMSEA and CFI, whereas unequal weights can be used to indicate a preference for one fit index over the other. If either weight is set to zero, the corresponding fit index has no effect on optimization. Thus, the optimization procedure seeks to find values of ν_{e} and ϵ such that the weighted sum of the mean squared error between the observed and target RMSEA and the mean squared error between the observed and target CFI is minimized.

The right-most term in Equation ((2.1)) consists of a user-defined penalty, λ, and an indicator function, $\mathbf{1}_{\boldsymbol{W}}$. The indicator function $\mathbf{1}_{\boldsymbol{W}}$ is equal to one whenever a user-specified number of (absolute) minor factor loadings are greater than some threshold value for any minor factor, and zero otherwise. The addition of this penalty term ensures a clear delineation between major and minor factors. For instance, a user could specify that no more than two minor factor loadings should be greater than .3 in absolute value. If the penalty term is set to some large value (e.g., $\lambda = 100$), TKL parameters that lead to models with too many large minor factor loadings will be heavily penalized and therefore excluded as candidates.

Unfortunately, empirical testing of the multiple-target method has shown that some combinations of target fit index values and constraints on the $\mathbf{W}$ matrix can sometimes lead to non-convergence when using the L-BFGS-B algorithm. Non-convergence can often be remedied by restarting optimization with different starting parameter values. If non-convergence still occurs after trying multiple different starting values, global optimization using a genetic algorithm (GA; Holland, 1975) can be used to minimize the objective function in Equation ((2.1)).

GAs take inspiration from biological evolutionary processes and natural selection to find candidate solutions that have a high level of fitness (i.e., that lead to high/low objective function values during maximization/minimization). Many variations of GAs have been proposed (Scrucca, 2013), but the procedure for most GAs can be generally described as follows. First, a random set of candidate solutions is generated and the fitness (i.e., objective function value) for each solution is evaluated. Next, a pair of "parent" solutions are selected (with replacement) from the set such that solutions with higher fitness have a higher selection probability. The parent solutions then produce two offspring with user-defined crossover and mutation probabilities. If crossover occurs, the children are formed as combinations of their parents using some crossover function. If crossover does not occur, the children are formed as exact copies of their parents. Similarly, if mutation occurs, the child solution is randomly perturbed. This process continues until there are as many children in the new generation as there were parents in the previous generation. Once the new generation is formed, selection occurs again and the process continues until a fixed number of generations have passed or until some stopping criterion is reached (Mitchell, 1996; Scrucca, 2013).

Although GAs work well for many problems where derivative-based methods have difficulty (e.g., when the objective function is not smooth or when there are local optima; Scrucca, 2013), a downside is that they can be relatively slow compared to derivative-based optimization methods when applied to more "well-behaved" optimization problems. Therefore, my optimization procedure first attempts to use the L-BFGS-B method to find a solution (with multiple random starts if convergence does not occur). If convergence still does not occur after multiple random starts using L-BFGS-B, a GA is used to find a solution. The advantage of this approach is that the procedure quickly produces a solution when the user-specified values make the problem well-suited for derivative-based optimization. If the derivative-based optimization fails, the procedure can still find a solution using the GA, albeit somewhat more slowly.

It is important to note that the multiple-target method will not necessarily produce solutions with RMSEA and CFI values that are exactly equal to the target values. For some models, it can be difficult to obtain particular combinations of RMSEA and CFI values (e.g., when modeling correlation matrices with many items, a low ϵ_T value and a high CFI_T value). Moreover, the method is not guaranteed to find solutions that are global minima. Thus, users should check solutions to make sure that the model fit values are sufficiently close to the target values for their purposes. These issues will be investigated more thoroughly in the simulation study that is described in the next section.

Chapter 3

Methods

3.1 Simulation Design

I conducted a simulation study to investigate and compare characteristics of the TKL, CB, and WB model-error methods and to evaluate the effectiveness of the multiple-target TKL method. Two questions related to aspects of the model-error generation process were of particular interest. First, how do the model-error methods differ in terms of the characteristics of the error-perturbed covariance matrices they produce? Specifically, how do the model-error methods differ in terms of model fit statistics (e.g., RMSEA, SRMR, CFI, TLI) for the error-perturbed covariance matrices they generate? Second, how well do the model-error methods (in particular, the multiple-target method) work in terms of producing error-perturbed covariance matrices with RMSEA (and CFI) values that are close to the target values? Answering these questions should be helpful to researchers who are planning Monte Carlo simulation studies involving covariance structure models and would like to understand how their choice of model-error method is likely to affect the characteristics of simulated, error-perturbed covariance matrices.

In the simulation study, I compared model-error methods by generating error-perturbed covariance matrices using a variety of population models and comparing the results based on several model fit indices. For simplicity, I used covariance matrices with unit diagonals (i.e., correlation matrices). Moreover, I focused on four model fit indices: the RMSEA, the CFI, the Tucker-Lewis Index (TLI; Tucker & Lewis, 1973), and the Correlation Mean Squared Residual (CRMR; K. A. Bollen, 1989; Ogasawara, 2001). Although many other fit indices have been proposed (see Marsh et al., 2005), the selected fit indices are among the most commonly-used and include both measures of absolute fit (RMSEA, CRMR) and incremental fit (CFI, TLI).

Because the relationship between fit indices is affected by model characteristics (Lai & Green, 2016), I included a variety of distinct population models created by systematically varying: (a) the number of major factors (Factors $\in \{1, 3, 5, 10\}$), (b) the number of items per factor (Items/Factor $\in \{5, 15\}$), (c) the correlation between factors ($\phi \in \{0, .3, .6\}$), and (d) the strength of the item factor loadings (Loadings $\in \{0.4, 0.6, 0.8\}$). Each item loaded on only a single factor and factor loadings were fixed at values representing weak, moderate, and strong factor loadings, respectively (Hair et al., 2018). These factor loading and factor correlation values were intended to represent a range of values representative of values observed in empirical research. For instance, in a confirmatory factor analysis of subtests from the Ball Aptitude Battery believed to measure aspects of intelligence, Neuman et al. (2000) reported estimated factor loadings between 0.26 and 0.95 and factor correlations between .18 and .73.

Forming a fully-crossed design from the levels of Factors, Items/Factor, and factor correlation (ϕ) would have resulted in 4 (Factors) $\times$ 2 (Items/Factor) $\times$ 3 (ϕ) $\times$ 3 (Loadings) = 72 unique conditions. However, the 12 conditions with one factor and factor correlations greater than zero were invalid because it is nonsensical to have correlated factors for one-factor models. For the 60 remaining conditions, I computed the model-implied population correlation matrix corresponding to the population common factor model indicated by the condition. To generate model-implied correlation matrices (without model error), I used the `simFA()` function in the R *fungible* library (Waller, 2021). The `simFA()` function computes population correlation matrices for common factor models by taking the model parameters (e.g., factor loadings, number of items per factor, factor correlations) as arguments and

Table 3.1
Matrices of factor loadings corresponding to the (a) Weak, (b) Moderate, and (c) Strong factor loading conditions.

(a)

F1	F2	F3
0.4	0.0	0.0
0.4	0.0	0.0
0.4	0.0	0.0
0.4	0.0	0.0
0.4	0.0	0.0
0.0	**0.4**	0.0
0.0	**0.4**	0.0
0.0	**0.4**	0.0
0.0	**0.4**	0.0
0.0	**0.4**	0.0
0.0	0.0	**0.4**
0.0	0.0	**0.4**
0.0	0.0	**0.4**
0.0	0.0	**0.4**
0.0	0.0	**0.4**

(b)

F1	F2	F3
0.6	0.0	0.0
0.6	0.0	0.0
0.6	0.0	0.0
0.6	0.0	0.0
0.6	0.0	0.0
0.0	**0.6**	0.0
0.0	**0.6**	0.0
0.0	**0.6**	0.0
0.0	**0.6**	0.0
0.0	**0.6**	0.0
0.0	0.0	**0.6**
0.0	0.0	**0.6**
0.0	0.0	**0.6**
0.0	0.0	**0.6**
0.0	0.0	**0.6**

(c)

F1	F2	F3
0.8	0.0	0.0
0.8	0.0	0.0
0.8	0.0	0.0
0.8	0.0	0.0
0.8	0.0	0.0
0.0	**0.8**	0.0
0.0	**0.8**	0.0
0.0	**0.8**	0.0
0.0	**0.8**	0.0
0.0	**0.8**	0.0
0.0	0.0	**0.8**
0.0	0.0	**0.8**
0.0	0.0	**0.8**
0.0	0.0	**0.8**
0.0	0.0	**0.8**

then using the equation for the common factor model (i.e., Equation ((1.1))) to produce the population correlation matrix corresponding to the specified model.

Having generated model-implied population correlation matrices, the next step in the simulation procedure was to generate population correlation matrices with model error for each of the 60 population factor models using the multiple-target TKL, the CB, and the WB model-error methods. Each model-error method was repeated with random starting conditions 500 times for each of three target RMSEA values ($\varepsilon_T \in \{0.025, 0.065, 0.090\}$) and each of the 60 population factor models. The target RMSEA values were chosen to represent models with very good, fair, and poor model fit, following the convention used by Myers et al. (2015) and MacCallum et al. (2001).

Although the TKL method has so far been discussed as a single model-error method, several variations of the multiple-target TKL method were included in the simulation study. Specifically, I generated error-perturbed covariance matrices for each condition with the multiple-target method using (a) only target RMSEA values (denoted as $\text{TKL}_{\text{RMSEA}}$), (b) equally-weighted target RMSEA and CFI values ($\text{TKL}_{\text{RMSEA/CB}}$), and (c) only target CFI values (TKL_{CB}). I used target CFI values (denoted as CFI_T) corresponding to the same subjective levels of model fit as the previously-mentioned target RMSEA values, forming conditions with very good ($\text{RMSEA}_T = 0.025$, $\text{CFI}_T = 0.99$), fair ($\text{RMSEA}_T = 0.065$, $\text{CFI}_T = 0.95$), and poor ($\text{RMSEA}_T = 0.090$, $\text{CFI}_T = 0.90$) model fit (Hu & Bentler, 1999). Additionally, each of the multiple-target TKL methods included a penalty term, $\lambda = 1,000,000$, to penalize solutions where any minor common factor had more than two loadings greater than .3.

For all of the multiple-target TKL methods, the L-BFGS-B optimization procedure was restarted with new starting values of ν_{e} and if it failed to converge within 1,000 iterations. Starting values were randomly generated using $\nu_{\text{e}0} \sim \mathcal{U}(.2, .9)$ and $\epsilon_0 \sim \mathcal{U}(0, .8)$, where $\nu_{\text{e}0}$ and ϵ_0 denote the starting values of ν_{e} and ϵ and $\mathcal{U}(a, b)$ denotes a uniform distribution on the interval $[a, b]$. These distributions were chosen because initial testing indicated that the multiple-target TKL method was more likely to result in a converged solution if the range of the starting values were somewhat restricted.

In addition to randomly initializing the starting values of ν_{e} and ϵ, the values of the $\mathbf{W}^*$ matrix were also randomly initialized at each repetition. In the multiple-target TKL method, the $p \times q$ provisional matrix $\mathbf{W}^*$ was initialized such that each column consisted of p samples from a standard normal distribution. Let ϵ_j and $\nu_{\text{e}j}$ denote the proposed values of ϵ and ν_{e} at iteration j of the multiple-target TKL optimization procedure. At each jth iteration, the columns of the $\mathbf{W}^*$ were scaled using $\mathbf{W}^*_j = \mathbf{W}^*\mathbf{V}$, where $\mathbf{V}$ denotes a $q \times q$ diagonal matrix with diagonal elements $(1 - \epsilon_j)^0, (1 - \epsilon_j)^1, \ldots, (1 - \epsilon_j)^q$. Then, $\mathbf{W}^*_j$ was scaled as described in Section @ref(#tkl-method) to form a $\mathbf{W}_j$ matrix to ensure that the proportion of variance accounted for by the q minor common factors was equal to $\nu_{\text{e}j}$.

Repeating each of the TKL method variations 500 times with random starting values was important because the multiple-target TKL method is not guaranteed to find the global optimum for a particular problem. Moreover, the results of the multiple-target TKL method are affected by the particular value of $\mathbf{W}^*$. Therefore, looking at the results of the multiple-target TKL method with a single set of starting values and a single $\mathbf{W}^*$ matrix might have led to results that were somewhat idiosyncratic and not representative of the method's general performance. It might reasonably be asked why $\mathbf{W}^*$ was not held fixed over the 500 repetitions with random starting values to find the global optimum parameter values *for that particular* $\mathbf{W}^*$ *matrix*. However, the purpose of the multiple-target TKL method was to produce solutions with fit index values that are "close enough" to the target values to be reasonable over the entire space of $\mathbf{W}^*$ matrices, not to find the best solution for one particular $\mathbf{W}^*$ matrix.

As with the multiple-target TKL methods, the WB method was also repeated 500 times for each condition of the simulation design. Each repetition of the WB method represented an independent sample from the inverse Wishart distribution associated with each condition. Therefore, a large number of samples were required to ensure that the simulation results for the WB method represented the full range of potential outcomes.

[ROUGH TRANSITION] Additional clarification is needed concerning the implementation of the WB method. Unlike the CB model-error method, the WB model-error method does not allow precise control of RMSEA values and produces RMSEA values that are only approximately equal to the target value. Simply using a target RMSEA value to solve for a value of v was unlikely to result in error-perturbed covariance matrices that had RMSEA values close to the target RMSEA value, unless the target RMSEA value was relatively small. This is because $v = \varepsilon^2 + o_\mathrm{p}(\varepsilon^2)$ (Wu & Browne, 2015a), where $o_\mathrm{p}(\varepsilon^2)$ (read as "little-o-p-ε^2") indicates that the difference between v and ε^2 is bounded in probability at the rate ε^2 (Vaart, 1998, p. 12) [***COME BACK TO THIS; Wu and Browne p. 580***]. To resolve this issue, I developed a method to find a value of v such that the median RMSEA value from the resulting error-perturbed covariance matrices was close to the target RMSEA value. The method is as follows. First, initial values of v were obtained by squaring each element in a sequence of 20 equally-spaced RMSEA values ranging from 0.01 to 0.095.

For each v value, 50 error-perturbed covariance matrices were sampled from the inverse Wishart distribution and the median RMSEA value was computed. Next, v was regressed on the linear and squared median RMSEA terms. The fitted model was then used to predict an appropriate v value such that the median RMSEA value for simulated, error-perturbed covariance matrices was close to the target value.

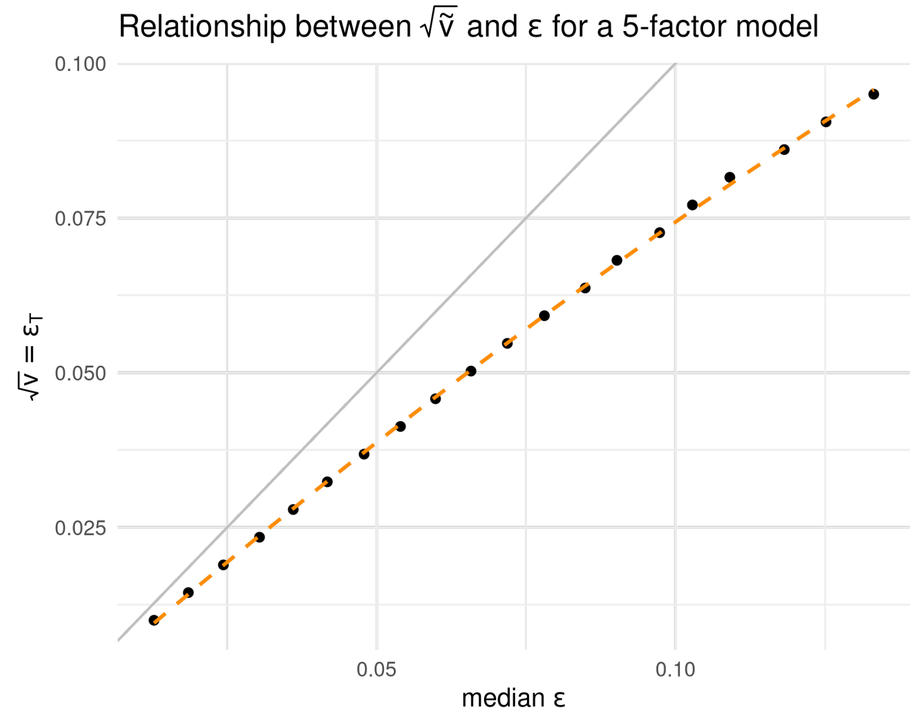

Figure 3.1. The solid gray line indicates where the target RMSEA and median RMSEA values (from 50 samples) would be equal. The dashed orange line indicates the predicted value of $\sqrt{v}$ that will produce a given RMSEA value.

In summary, the design of the simulation study was as follows. The crossed combinations of number of factors, number of items per factor, factor correlation, and factor loading configurations correspond to 60 population major factor models. For each of those population models, I generated 500 error-perturbed correlation matrices using three variations of the TKL multiple-target method, the CB method, and the WB method at each of three target model fit conditions corresponding to very good, fair, and poor model fit. In total, this will

Table 3.2
Simulation Study Design Variables and Levels.

Variable	Levels
Factors	1, 3, 5
Items/Factor	5, 15
Factor Correlation (ϕ)	0.0, 0.3, 0.6
Factor Loading	0.4 , 0.6, 0.8
Target Model Fit	Very Good ($\varepsilon_T = 0.025$, $\text{CFI}_T = 0.99$),
	Fair ($\varepsilon_T = 0.065$, $\text{CFI}_T = 0.95$),
	Poor ($\varepsilon_T = 0.090$, $\text{CFI}_T = 0.90$)
Model-Error Method	$\text{TKL}_{\text{RMSEA}}$, TKL_{CFI}, $\text{TKL}_{\text{RMSEA/CFI}}$, CB, WB

Note. ε_T = Target RMSEA value; CFI_T = Target CFI value. TKL subscripts indicate which model fit indices were used as targets.

result in 900 unique conditions and a total of 4.5×10^5 simulated error-perturbed correlation matrices. All of the independent variables in the study (and the levels of those variables) are summarized in Table Table 3.2.

3.2 Simulation Procedure

In the previous section, I described the 60 population models (without model error) resulting from the crossed levels of number of factors, number of items per factor, factor loadings, and factor correlations. In the first stage of the simulation study, I computed the model-implied population correlation matrices for each of the population models using the `simFA()` function in the R *fungible* library (Waller, 2021). The `simFA()` function computes population correlation matrices for common factor models by taking the model parameters (e.g., factor loadings, number of items per factor, factor correlations) as arguments and then using the equation for the common factor model (i.e., Equation ((1.1))) to produce the population correlation matrix corresponding to the specified model.

After computing model-implied population correlation matrices for each of the 60 population models, the next step in the simulation procedure was to generate population correlation matrices with model error using each of the model-error methods described in the previous section. For each of the $60 \times 3 = 180$ crossed combinations of population model and target model fit, I generated population correlation matrices with model error

1. TKL

Three variations of the multiple-target TKL method were used to generate population correlation matrices with model error: (TKL$_{\text{RMSEA}}$, TKL$_{\text{CFI}}$, and TKL$_{\text{RMSEA/CFI}}$. For each of these methods, 500 population correlation matrices with model error were generated using random starting values $\nu_{\text{e}0} \sim \mathcal{U}(.2, .9)$ and $\epsilon_0 \sim \mathcal{U}(0, .8)$, where $\nu_{\text{e}0}$ and ϵ_0 denote the starting values of ν_{e} and ϵ and $\mathcal{U}(a, b)$ denotes a uniform distribution on the interval $[a, b]$. These distributions were chosen because initial testing indicated that the multiple-target TKL method was more likely to result in a converged solution if the range of the starting values were somewhat restricted.

In addition to randomly initializing the starting values of ν_{e} and ϵ, the values of the $\mathbf{W}^*$ matrix were also randomly initialized at each repetition. In the multiple-target TKL method, the $p \times q$ provisional matrix $\mathbf{W}^*$ was initialized such that each column consisted of p samples from a standard normal distribution. Let ϵ_j and $\nu_{\text{e}j}$ denote the proposed values of ϵ and ν_{e} at iteration j of the multiple-target TKL optimization procedure. At each jth iteration, the columns of the $\mathbf{W}^*$ were scaled using $\mathbf{W}^*_j = \mathbf{W}^*\mathbf{V}$, where $\mathbf{V}$ denotes a $q \times q$ diagonal matrix with diagonal elements $(1-\epsilon_j)^0, (1-\epsilon_j)^1, \ldots, (1-\epsilon_j)^q$. Then, $\mathbf{W}^*_j$ was scaled as described in Section Section 1.1.1 to form a $\mathbf{W}_j$ matrix to ensure that the proportion of variance accounted for by the q minor common factors was equal to $\nu_{\text{e}j}$.

Repeating each of the TKL method variations 500 times with random starting values was important because the multiple-target TKL method is not guaranteed to find the global optimum for a particular problem. Moreover, the results of the multiple-target TKL method are affected by the particular value of $\mathbf{W}^*$. Therefore, looking at the results of the multiple-target TKL method with a single set of starting values and a single $\mathbf{W}^*$ matrix might have led to results that were somewhat idiosyncratic and not representative of the method's general performance.

It might reasonably be asked why $\mathbf{W}^*$ was not held fixed over the 500 repetitions with random starting values to find the global optimum parameter values *for that particular* $\mathbf{W}^*$ *matrix*. However, the purpose of the multiple-target TKL method is to produce solutions with fit index values that are "close enough" to the target values to be reasonable over the entire space of $\mathbf{W}^*$ matrices, not to find the best solution for one particular $\mathbf{W}^*$ matrix.

As with the multiple-target TKL methods, the WB method was also repeated 500 times for each condition of the simulation design. Each repetition of the WB method represented an independent sample from the inverse Wishart distribution associated with each condition. Unlike the multiple-target TKL methods and the WB method, the implementation of the CB method that was used was deterministic. That is, given the same population model, the CB method always returned the same correlation matrix with model error. Therefore, it did not make sense to conduct 500 repetitions of the CB method for each condition because every repetition would have given the exact same result [NEED TO MAKE SURE PREVIOUS METHODS SECTION REFLECTS THIS].

Moreover, R code for the simulation study (including implementations of each of the model-error methods) can be found in Appendix A [hyperlink this].

Chapter 4

Results

In the previous section, I described the simulation study I conducted to learn more about the behavior of different methods for generating error-perturbed population covariance (correlation) matrices. The simulation study included three model-error methods—the (single- and multiple-target) TKL method, the CB method, and the WB method—and was designed to answer two primary questions.

First, I wanted to know whether different model-error methods led to different values of the CFI, TLI, and CRMR fit indices when used with the same error-free models and target RMSEA values. If the model-error methods led to systematically different values on the alternative fit indices when matched on RMSEA and all other characteristics, it would suggest that they are not exchangeable. In that case, researchers conducting simulation studies would have to consider which of the model-error methods most closely approximates the model error process they are trying to simulate. Moreover, such results would highlight the importance of reporting fit indices other than RMSEA when simulating imperfect models. If there were no meaningful differences among the methods, it would indicate that the choice of one particular model-error method over another (among the methods considered here) is not an important variable in the design of simulation studies.

A second purpose of the study was to evaluate the effectiveness of the proposed multiple-target method for generating correlation matrices with model error that had RMSEA and

CFI values that were close to the specified target values. It was not expected that the algorithm would be able to produce correlation matrices with RMSEA and CFI values that were very close to the target values for all of the major-factor population models because of the relationship between RMSEA, CFI, and population model characteristics (Lai & Green, 2016). Therefore, I used the absolute deviation between the observed and target RMSEA and CFI values to compare the results from the multiple-target TKL method to the results from the CB and WB methods used in Study 1.

The remainder of the section is structured as follows. First, I report how many of the simulated matrices had properties that would make them unsuitable for use in a simulation study. XXX...

4.1 Indefinite Matrices (CB)

One drawback of the CB model-error method is that the resulting correlation matrix with model error can be indefinite (i.e. having one or more negative eigenvalues) when the specified target RMSEA value is large (Cudeck & Browne, 1992). These matrices are undesirable because correlation and covariance matrices are, by definition, at least positive semi definite (i.e., having strictly non-negative eigenvalues). Of the 90000 correlation matrices with model error that were generated using the CB method, 14291 (15.9%) were indefinite. However, indefinite solutions were much more common for some conditions of the simulation design than other. Figure Figure 4.1 shows the percent of indefinite CB solutions for each level of model fit, number of items per factor, number of factors, and factor loading strength are shown in Figure Figure 4.1. (Exact percentages are reported in Table Table 4.1). The figure shows at least three notable trends. First, the percent of indefinite solutions increased as model fit degraded. Second, the percent of indefinite solutions increased as the total number of items increased (i.e., as the number of factors and items per factor increased). Finally, Figure Figure 4.1 shows that the percent of indefinite solutions increased as factor loadings increased.

In the best-case scenarios, conditions corresponding to models with 25 items or fewer led to indefinite solutions very infrequently (in less than 1% of cases). On the other hand, conditions with Poor model fit and 45 items or more led to indefinite correlation matrices in more than 90% of cases. These results show that the CB method would be an inefficient way to simulate positive semi definite population correlation or covariance matrices with model error when input matrices are large and the target RMSEA value is relatively large.

Although inefficient, a potential strategy for dealing with indefinite solutions when using the CB method is to simply generate solutions using the CB method until a sufficient number of positive semi definite solutions have been obtained. However, the amount of time taken by the CB method increases quickly as the number of items increase, making the oversampling strategy impractical for problems with many items. In fact, using the CB method to generate even a small number of solution matrices becomes impractical for large input matrices. This is demonstrated in Figure Figure 4.2, which shows the completion time for the CB method when applied to a one-factor model with salient loadings fixed at .6 and the number of items varying between 5 and 120. Using a computer with an Intel Core i5-4570 3.20GHz CPU and 16GB of RAM, the CB method took just over 30 seconds to complete (on average) for an input correlation matrix with 65 items. For an input matrix with 115 items, the CB method took approximately four and a half minutes to complete. Long completion times are often impractical for large simulation studies, particularly if indefinite solutions are discarded. In fact, I had to skip using the CB method in simulation conditions with 10 factors and 15 items per factor (150 items) because those conditions would have taken an impractical amount of time to complete. Timing a single example, the CB method took 15 minutes and 48 seconds to complete with a 150-item input correlation matrix. At that rate, it would have taken almost 11 days to complete one (out of 36) conditions of the simulation design with 150 items.

Table 4.1
The percent of Cudeck and Browne (CB) model-error method solutions that were indefinite.

			Factors			
Items/Factor	Loading	Model Fit	1	3	5	10
5	0.4	Very Good	0.0	0.0	0.0	0.0
5	0.4	Fair	0.0	0.3	0.1	0.0
5	0.4	Poor	0.0	0.1	1.1	0.3
5	0.6	Very Good	0.0	0.0	0.0	0.0
5	0.6	Fair	0.0	0.0	0.7	0.0
5	0.6	Poor	0.2	0.4	0.9	0.5
5	0.8	Very Good	0.0	0.0	0.0	0.0
5	0.8	Fair	0.0	0.1	0.1	0.0
5	0.8	Poor	0.0	0.7	1.3	0.7
15	0.4	Very Good	0.0	0.0	0.3	—
15	0.4	Fair	0.2	1.7	68.7	—
15	0.4	Poor	0.4	91.7	99.1	—
15	0.6	Very Good	0.0	0.0	0.0	—
15	0.6	Fair	0.2	2.2	75.1	—
15	0.6	Poor	1.4	91.2	98.9	—
15	0.8	Very Good	0.0	0.0	0.1	—
15	0.8	Fair	0.2	1.7	76.5	—
15	0.8	Poor	0.4	93.4	99.0	—

Note. The Cudeck-Browne method was not used for conditions with 10 major factors and 15 items per factor because it was prohibitively slow for those conditions.

```
‘summarise()‘ has grouped output by ’factors’, ’items_per_factor’,

’loading_numeric’. You can override using the ‘.groups‘ argument.
```

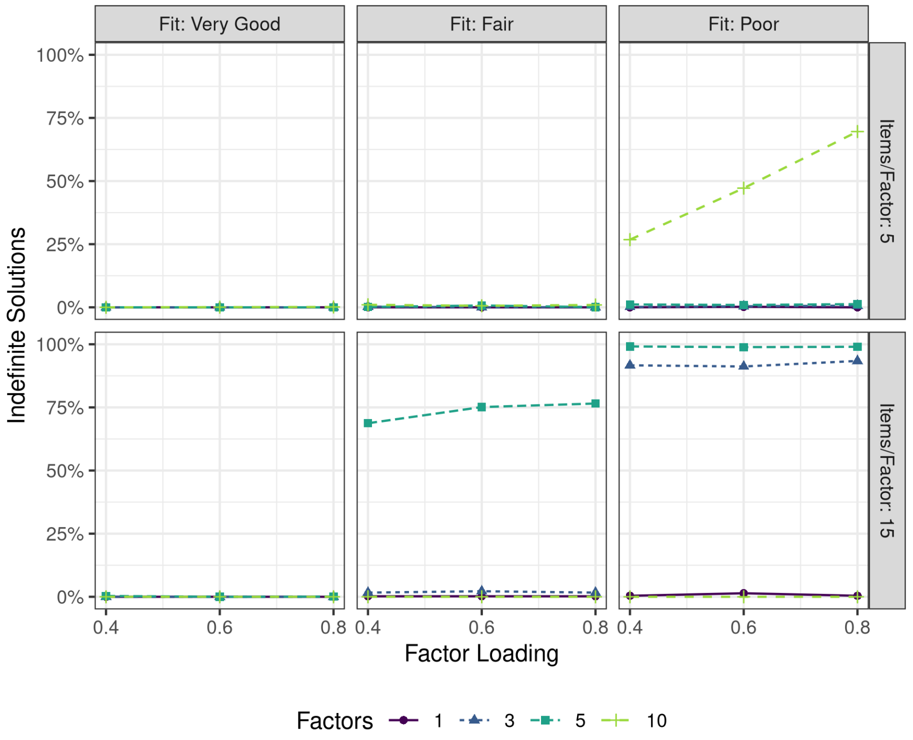

Figure 4.1. The percent of Cudeck-Browne (CB) method solutions that were indefinite, conditioned on number of factors, factor loading, number of items per factor, and model fit.

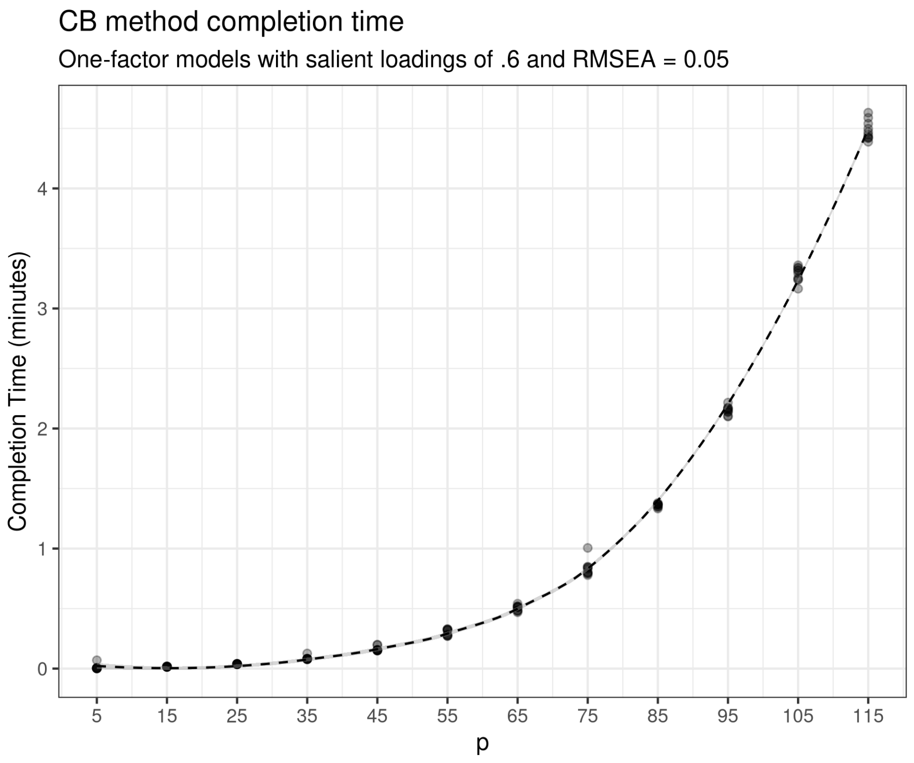

Figure 4.2. The amount of time (in minutes) taken to generate a single correlation matrix with model error using the CB method. Completion times were recorded 10 times for single-factor models with salient factor loadings fixed at .6 and number of items (*p*) varying between 5 and 115. The dashed black line is the loess regression line.

4.2 L-BFGS-B Non-convergence (Genetic Algorithm)

The default optimization method for the multiple-target TKL method was L-BFGS-B. In most cases, this method worked well and converged relatively quickly to a solution. However, there were a small number of cases where the L-BGFS-B method failed to converge. Specifically, the L-BFGS-B method failed to converge 14 times ($< 1\%$) when the target CFI and RMSEA values were both used and when model fit was poor. The L-BFGS-B non-convergence rates for the $\text{TKL}_{\text{RMSEA/CFI}}$ method by number of factors, number of items per factor, and model fit are shown in Table Table 4.2. Non-convergence was also somewhat more likely for conditions with few factors compared to conditions with many factors.

Although non-convergence of the L-BFGS-B algorithm was rare, a question that arose was whether the genetic algorithm that was used as a fallback option led to similar results compared to cases where the L-BFGS-B algorithm converged. This question was difficult to answer statistically because non-convergence occurred so infrequently. To get a general sense of whether results for cases where the L-BFGS-B algorithm converged or did not converge were similar, I plotted the CFI and RMSEA values for all converged and non-converged cases in conditions where the multiple-target TKL method was used with target CFI and RMSEA values and model fit was Poor. Figure Figure 4.3 shows that cases where the L-BFGS-B algorithm did not converge (shown in black) led to similar RMSEA and CFI values compared to cases where the algorithm converged (shown in gray). Thus, using a genetic algorithm seemed to provide reasonable results in the few cases where the L-BFGS-B algorithm failed to converge (albeit much more slowly than the L-BFGS-B algorithm).

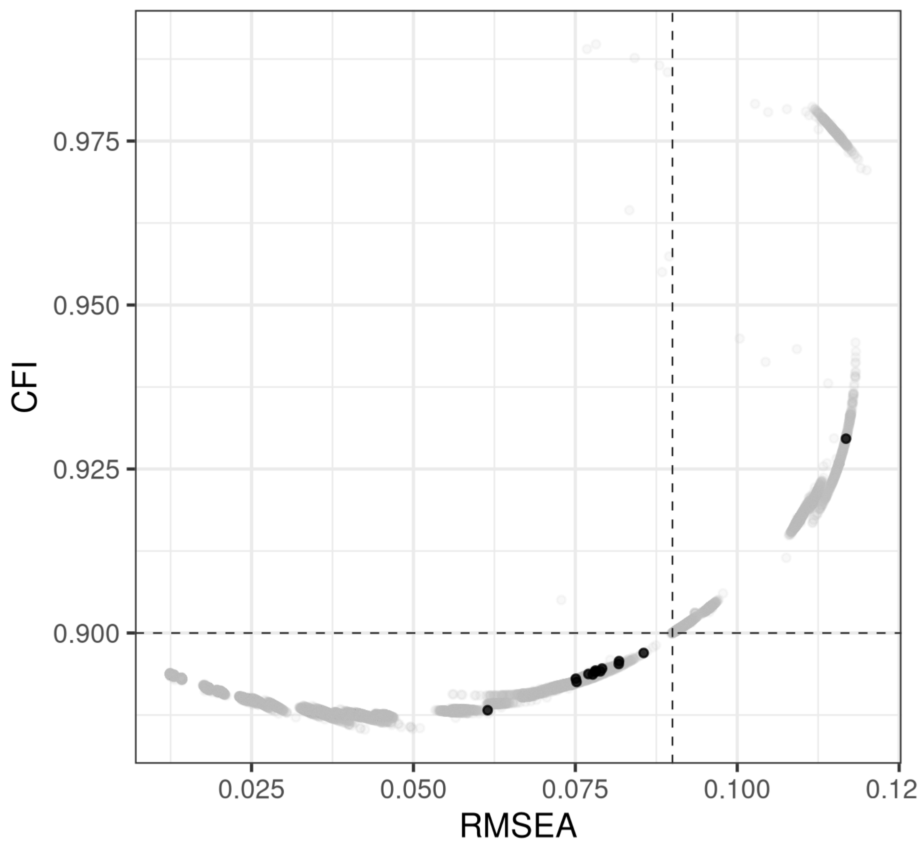

Figure 4.3. RMSEA and CFI values for cases where the model-error method was $\text{TKL}_{\text{RMSEA/CFI}}$ and model fit was Poor. Grey dots indicate cases where the L-BFGS-G algorithm converged; black dots indicate cases that did not converge and where a genetic algorithm was used instead.

```
‘summarise()‘ has grouped output by ’factors’, ’model_fit’. You can override

using the ‘.groups‘ argument.
```

Table 4.2
The percent of cases in each combination of conditions where the L-BFGS-B algorithm did not converge after 100 random starts and genetic optimization was used instead.

		Model Fit		
Factors	Items/Factor	Very Good	Fair	Poor
1	5	0.0	0.0	0.1
1	15	0.0	0.0	0.4
3	5	0.0	0.0	0.1
3	15	0.0	0.0	0.0
5	5	0.0	0.0	0.0
5	15	0.0	0.0	0.0
10	5	0.0	0.0	0.0
10	15	0.0	0.0	0.0

4.3 Major Minor Factors (W Matrices)

Recall that the multiple-objective TKL method included an optional penalty that heavily penalized cases that had strong minor factors. Specifically, the penalty was applied if any minor factor had two or more absolute factor loadings greater than or equal to a specified value. The purpose of the penalty was to avoid introducing minor factors that might be more accurately characterized as major factors. To determine whether the penalty was effective at helping avoid major minor factors, I checked each of the minor factor loading ($\mathbf{W}$) matrices to determine whether any minor factor had two or more loadings greater than .3 (in absolute value).

The percent of cases where the constraints on $\mathbf{W}$ were violated for each level of number of factors, number of items per factor, factor loading strength, and model fit are shown in Figure Figure 4.4 and reported in Table Table 4.3. Only results for the $\text{TKL}_{\text{RMSEA}}$ were included because the $\text{TKL}_{\text{RMSEA/CFI}}$ and TKL_{CFI} error methods seldom led to solutions that violated the constraints on $\mathbf{W}$. (Only 24 out of 90,000 cases had violated $\mathbf{W}$ constraints for the $\text{TKL}_{\text{RMSEA/CFI}}$ and TKL_{CFI} methods combined). Figure Figure 4.4 shows that the $\mathbf{W}$ constraints were violated most often when model fit was Fair or Poor, factor loadings were relatively low, and there were many total items (i.e., many factors and items per factor).

To understand why certain conditions led to more **W** constraint violations than others, it is helpful to understand the relationship between those conditions and the TKL parameters (ϵ and ν_{e}). Figure Figure 4.5 shows the ϵ and ν_{e} values for each of the TKL model-error methods, conditioned on factor loading strength, number of factors, and number of items per factor. To conserve space, only conditions with Poor model fit and 1, 5, or 10 major factors were included in the figure. Overall, the figure shows that there was a trade-off between ϵ and ν_{e} such that higher values of ν_{e} were related to lower values of ϵ (and *vice versa*). Moreover, Figure Figure 4.5 shows that the distributions of ϵ and ν_{e} differed depending on which TKL variant was used. The differences between the error-method variants were largest when the there were many items (i.e., many items and many items per factor) and when factor loadings were relatively weak. Under those circumstances, the $\text{TKL}_{\text{RMSEA}}$ method led to higher values of ν_{e} than the $\text{TKL}_{\text{RMSEA/CFI}}$ or TKL_{CFI} methods. This effect suggests that higher values of ν_{e} were required to produce solutions with RMSEA values close to .09 when there are many items. On the other hand, the results for the $\text{TKL}_{\text{RMSEA/CFI}}$ and TKL_{CFI} methods indicated that much lower ν_{e} values were required to obtain CFI values close to .90 when there were many items.

The apparent trade-off between ϵ and ν_{e} values makes sense when you consider their roles in the TKL method. A high ν_{e} value indicated that much of the unique variance would be assigned to the minor common factors. If the value of ϵ was also high, it indicated that the first few minor factors would account for most of the minor factor variance. Therefore, the **W** constraints were more likely to be violated when both ϵ and ν_{e} were high and less likely to be violated if either parameter was low. in Figure Figure 4.5, which shows the values of ϵ and ν_{e} produced by the three TKL variants for conditions with Poor model fit, 10 factors, 15 items per factor, and factor loadings of .8. Each point (corresponding to a single case) was colored according to whether or not the **W** constraints were violated. The figure shows that the **W** constraints were violated when the values of ϵ and ν_{e} were both above some

Table 4.3
The percent of cases that violated the minor common factor loading constraints for each level of number of factors, items per factor, factor loading, and model fit when the TKL$_{RMSEA}$ method was used.

			Model Fit		
Factors	Items per Factor	Loading	Very Good	Fair	Poor
1	5	0.4	0.0	0.0	0.0
1	5	0.6	0.0	0.0	0.0
1	5	0.8	0.0	0.0	0.0
1	15	0.4	0.0	3.2	39.4
1	15	0.6	0.0	0.0	15.0
1	15	0.8	0.0	0.0	0.0
3	5	0.4	0.0	0.1	22.9
3	5	0.6	0.0	0.0	4.7
3	5	0.8	0.0	0.0	0.0
3	15	0.4	0.0	57.3	79.3
3	15	0.6	0.0	33.3	74.9
3	15	0.8	0.0	0.0	7.4
5	5	0.4	0.0	4.8	52.3
5	5	0.6	0.0	0.0	34.5
5	5	0.8	0.0	0.0	0.0
5	15	0.4	0.0	79.0	92.6
5	15	0.6	0.0	73.3	89.7
5	15	0.8	0.0	0.0	81.2
10	5	0.4	0.0	44.7	75.9
10	5	0.6	0.0	12.1	68.9
10	5	0.8	0.0	0.0	1.6
10	15	0.4	0.0	94.5	99.9
10	15	0.6	0.0	94.8	98.3
10	15	0.8	0.0	82.3	96.8

threshold, which only happened when the TKL$_{\text{RMSEA}}$ was used.

```
‘summarise()‘ has grouped output by ’factors’, ’items_per_factor’,

’loading_numeric’. You can override using the ‘.groups‘ argument.
```

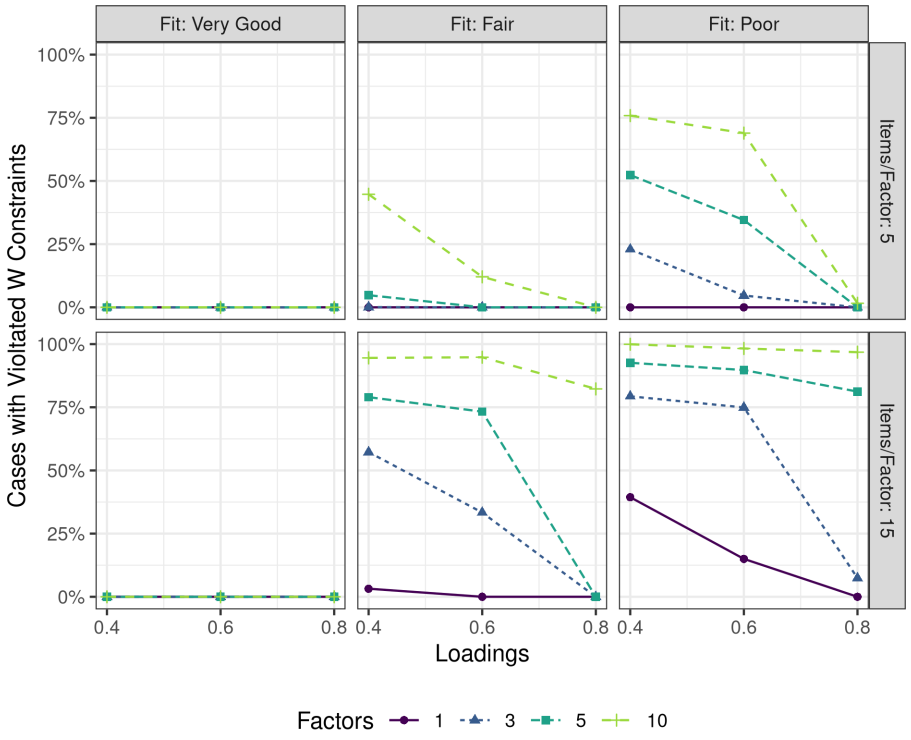

Figure 4.4. The percent of cases where the constraints on **W** were violated when the TKL$_{\text{RMSEA}}$ model-error method was used for each level of number of factors, number of items per factor, factor loading, and model fit.

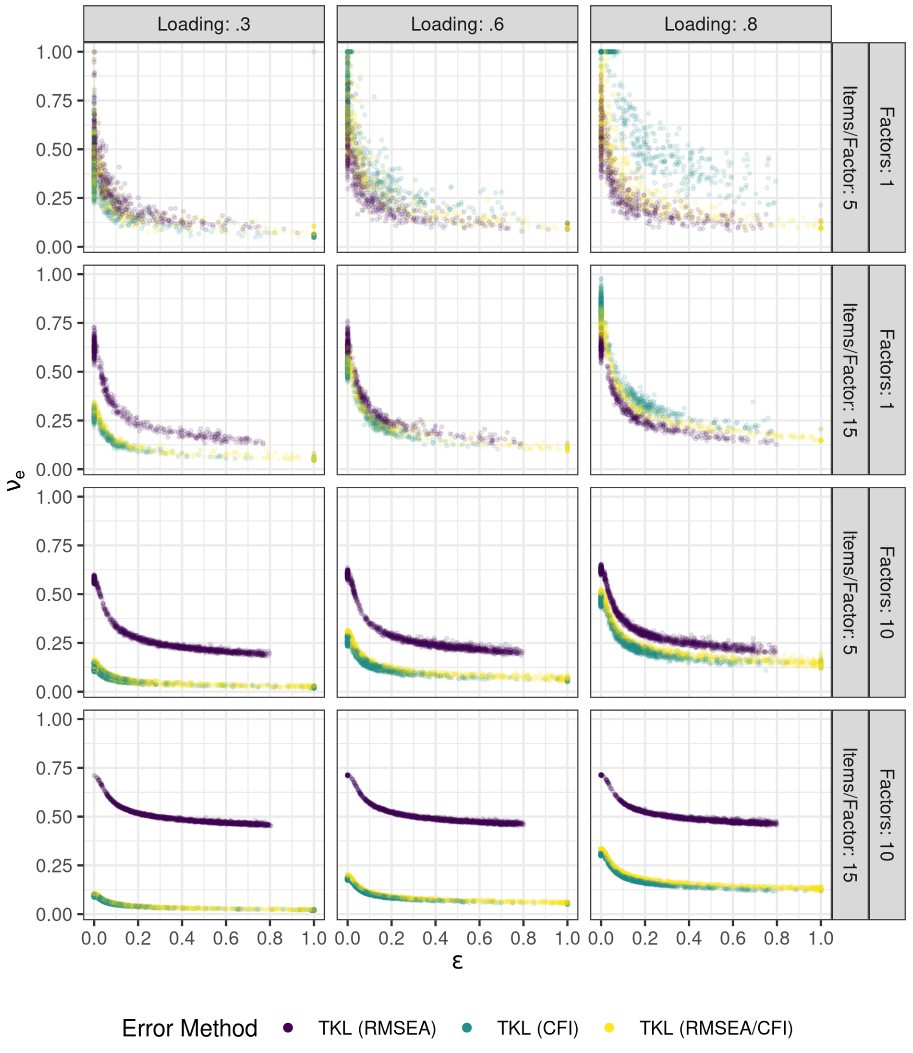

Figure 4.5. Values of the TKL parameters (ϵ and ν_{e}) by model-error method, number of factors, number of items per factor, and factor loading strength when model fit was Poor. Results for conditions with three or five major factors were omitted to conserve space. TKL = Tucker, Koopman, and Linn.

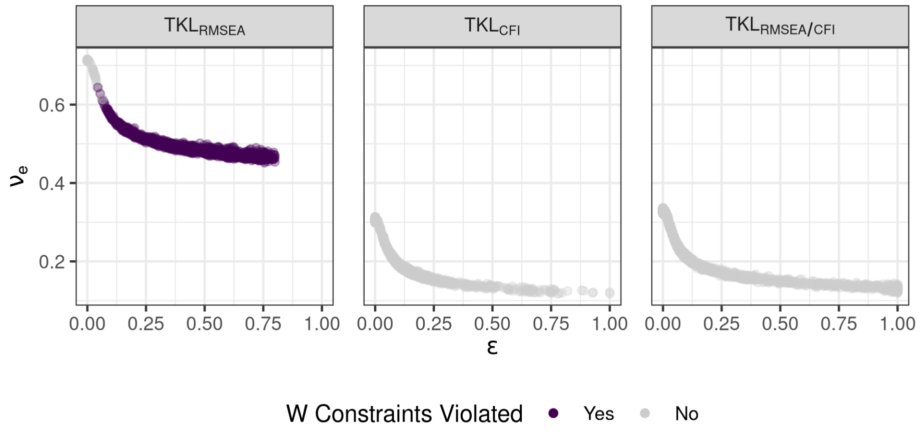

Figure 4.6. The distribution of ϵ and ν_e (and whether or not $\mathbf{W}$ constraints were violated) for conditions with Poor model fit, 10 factors, 15 items per factor, and factor loadings of .8. TKL = Tucker, Koopman, and Linn

4.4 Distributions of Fit Statistics

One of the primary questions the simulation study was intended to answer was whether the five model-error methods produced solutions with different fit index values when used with the same error-free models and target RMSEA and CFI values. In this section, I report the distributions of the RMSEA, CFI, TLI and CRMR model-fit indices for solutions produced by each of the five model-error methods ($\text{TKL}_{\text{RMSEA}}$, TKL_{CFI}, $\text{TKL}_{\text{RMSEA/CFI}}$, CB, and WB).

4.4.1 RMSEA

Of the fit indices investigated in this study, the RMSEA value has been most often used as a measure of model fit when generating covariance or correlation matrices with model error. Figure Figure 4.7 shows box-plots summarizing the distributions of RMSEA values for each of the model-error methods, conditioned on number of factors, model fit, and factor loading strength. Notice that the $\text{TKL}_{\text{RMSEA}}$ and CB methods almost always produced solutions with RMSEA values that were very close to the target RMSEA values. This result makes sense because both methods use optimization to produce solutions with RMSEA values close to a specified target. However, a somewhat unexpected result was that the CB method occasionally produced solutions with RMSEA values that were much higher or lower than the target value, particularly in conditions with ten major factors.

After the $\text{TKL}_{\text{RMSEA}}$ and CB methods, Figure Figure 4.7 shows that the WB method was the next best model-error method in terms of producing solutions with RMSEA values close to the target values. In fact, the WB method produced solutions with median RMSEA values that were as close to the target values as those from $\text{TKL}_{\text{RMSEA}}$ and CB solutions. However, the WB method also led to more variable RMSEA values, particularly in conditions with poor model fit and many factors.

The two methods that performed worst in terms of producing solutions with RMSEA values close to the targets were the $\text{TKL}_{\text{RMSEA/CB}}$ and TKL_{CB} methods. Figure Figure 4.7 shows that these methods often led to RMSEA values that were lower than the target values, except when there were relatively few factors and strong factor loadings. The largest differences between the observed and target RMSEA values for the $\text{TKL}_{\text{RMSEA/CB}}$ and TKL_{CB} methods occurred in conditions with poor model fit and relatively weak factor loadings (.3). In those conditions, both methods led to RMSEA values that were considerably lower than the target values.

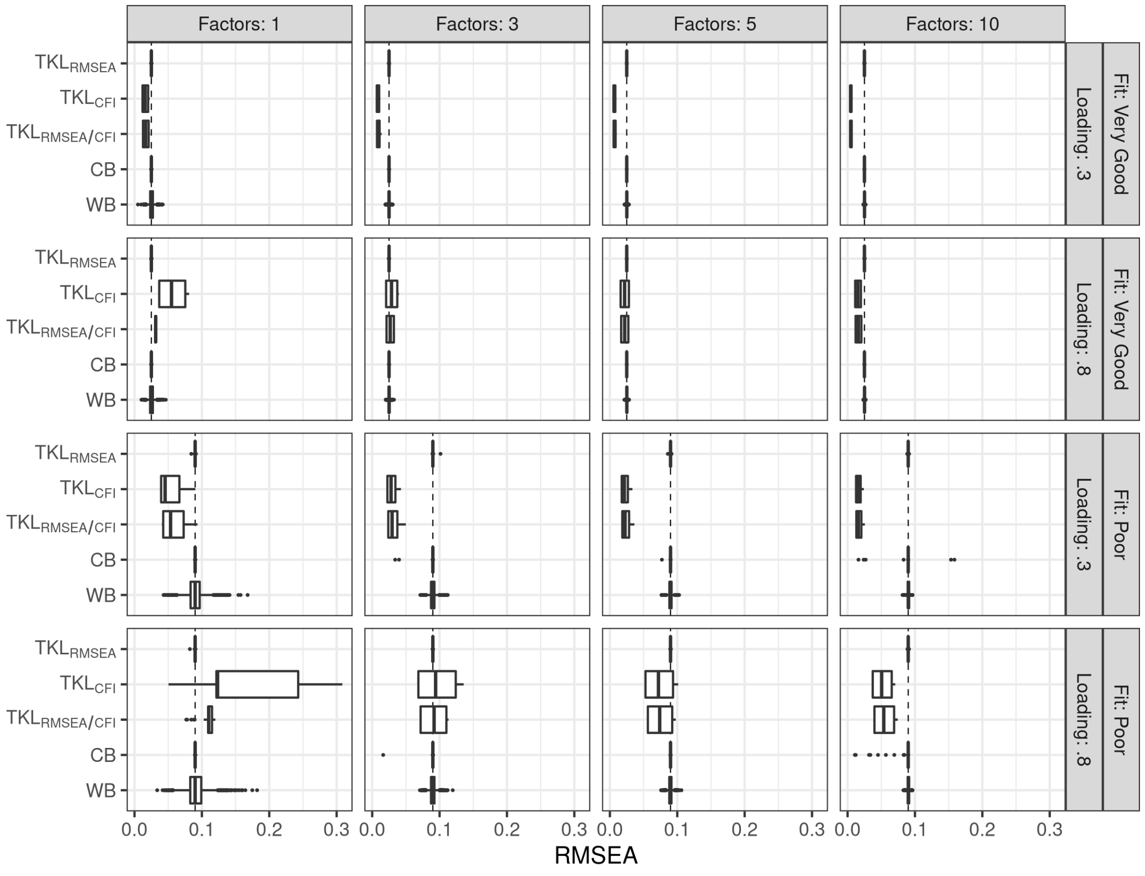

Figure 4.7. Distributions of the RMSEA values for solutions produced by each of the model-error methods, conditioned on number of factors, model fit, and factor loading strength. The dashed lines indicate the target RMSEA value for each condition. Note that some levels of model fit and factor loading strength were omitted to conserve space. TKL = Tucker, Koopman, and Linn; CB = Cudeck and Browne; WB = Wu and Browne.

4.4.2 CFI

[What are we doing for a transition here?] The distributions of CFI values for the solutions produced using the five model-error methods are shown in Figure Figure 4.8, conditioned on number of factors, model fit, and factor loading strength. As with Figure 4.7, the middle levels of model fit and factor loading strength were omitted to converse space. Figure Figure 4.8 shows that the results for CFI were nearly opposite to the results for RMSEA. Specifically, whereas the TKL$_{\text{RMSEA}}$, CB, and WB methods produced solutions with RMSEA values much closer to the target values than those produced by the $\text{TKL}_{\text{RMSEA/CFI}}$ or TKL_{CFI} methods, the $\text{TKL}_{\text{RMSEA/CFI}}$ and TKL_{CFI} produced solutions with CFI values that were closer to the target CFI values compared to the other model-error methods in most conditions. The differences between the methods that optimized for CFI ($\text{TKL}_{\text{RMSEA/CFI}}$ and TKL_{CFI}) and the other model-error methods were largest for conditions with poor model fit, poor factor loadings, and many factors (see the third row of Figure Figure 4.8). On the other hand, Figure Figure 4.8 shows that all of the model-error methods produced similar CFI values (that were close to the target CFI value) for conditions with very good model fit and strong factor loadings.

A result in Figure Figure 4.8 that is worth highlighting is that the $\text{TKL}_{\text{RMSEA/CFI}}$ and TKL_{CFI} produced solutions with very similar CFI values in most conditions. These two methods also produced solutions with very similar RMSEA values in many conditions, as shown in Figure Figure 4.7. Indeed, the $\text{TKL}_{\text{RMSEA/CFI}}$ and TKL_{CFI} methods led to much more similar results in terms of both RMSEA and CFI than the $\text{TKL}_{\text{RMSEA/CFI}}$ and $\text{TKL}_{\text{RMSEA}}$ methods. Put another way, optimizing for CFI alone generally led to similar results compared to optimizing for both CFI and RMSEA, whereas optimizing for RMSEA alone generally led to solutions with much different RMSEA and CFI values compared to optimizing for both CFI and RMSEA. This suggests that CFI was more sensitive to small changes in parameter values than RMSEA.

The distributions of CFI values shown in Figure Figure 4.8 also emphasize the importance of reporting multiple fit indices when simulating correlation or covariance matrices with model error. For instance, the $\text{TKL}_{\text{RMSEA}}$ and CB methods produced solutions with RMSEA values close to the target RMSEA value of .09 in conditions with poor model fit and weak factor loadings. However, Figure Figure 4.8 shows that those two model-error methods led to solutions with unacceptably low CFI values in the same conditions.

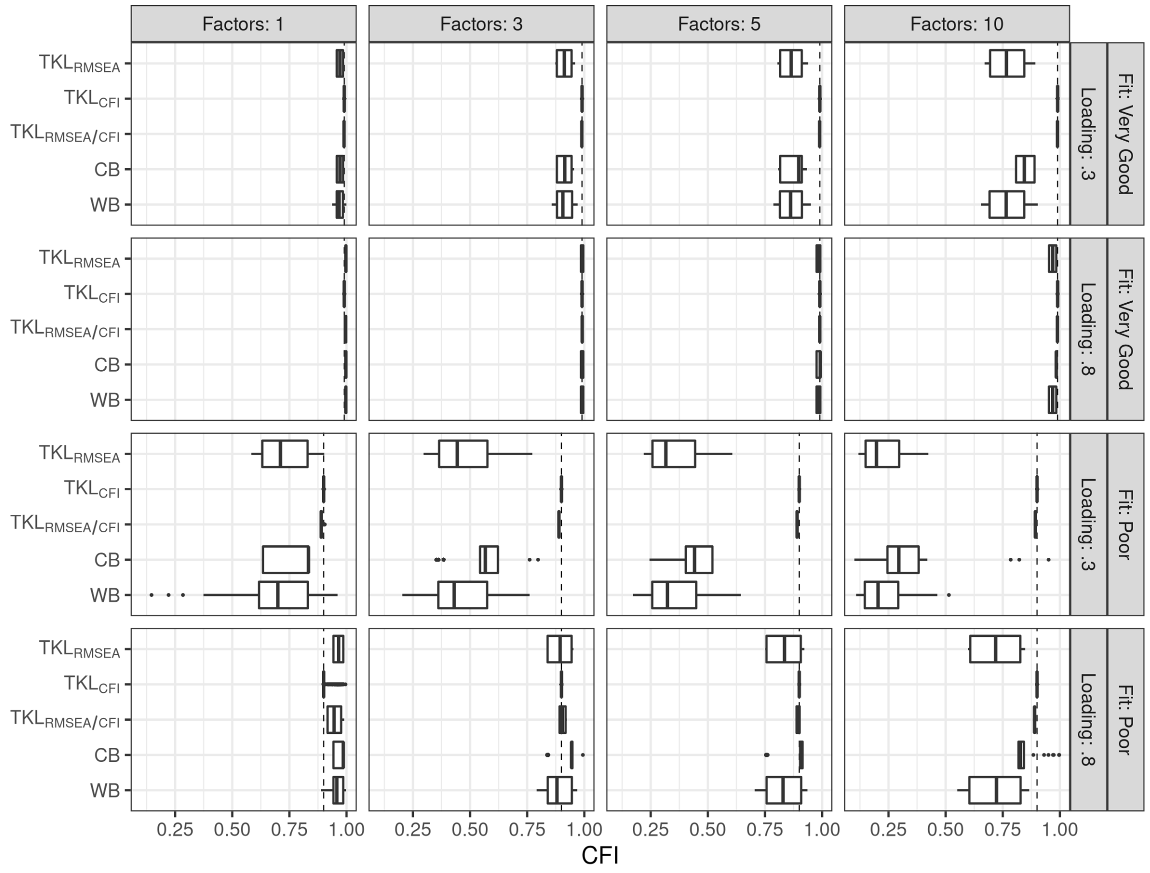

Figure 4.8. Distributions of the CFI values for solutions produced by each of the model-error methods, conditioned on number of factors, model fit, and factor loading strength. The dashed lines indicate the target CFI value for each condition. Note that some levels of model fit and factor loading strength were omitted to conserve space. TKL = Tucker, Koopman, and Linn; CB = Cudeck and Browne; WB = Wu and Browne.

4.4.3 TLI

The distributions of TLI values for the solutions produced using the five model-error methods are shown in Figure Figure 4.9, conditioned on number of factors, model fit, and factor loading strength. Overall, the distributions of TLI values were quite similar to the distributions of CFI values shown in Figure Figure 4.8. In particular, the TKL$_{\text{CFI}}$ and TKL$_{\text{RMSEA/CFI}}$ methods tended to produce solutions with higher TLI values than the other model-error methods, except for conditions with a single factor, Poor model fit, and strong factor loadings.

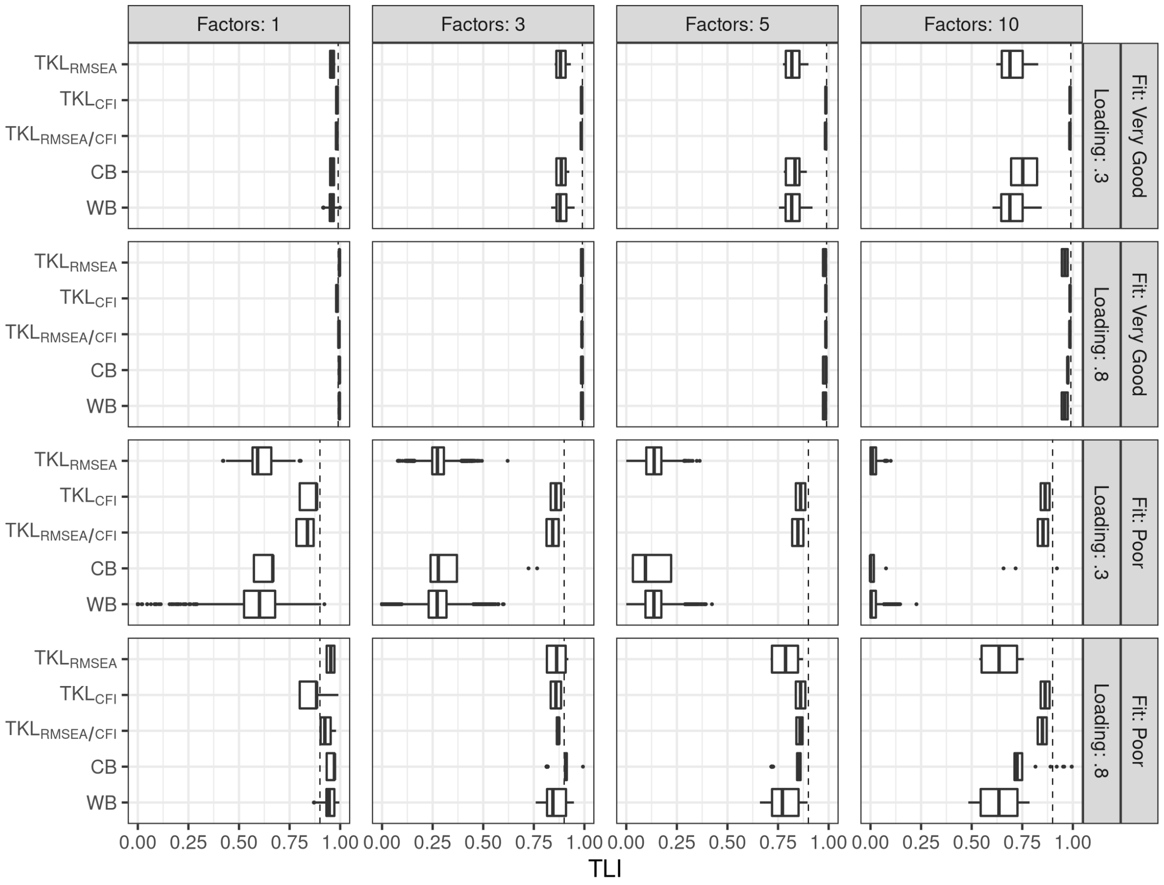

Figure 4.9. Distributions of the TLI values for solutions produced by each of the model-error methods, conditioned on number of factors, model fit, and factor loading strength. The dashed lines indicate the threshold values of TLI that correspond to the targeted levels of model fit, according to Hu and Bentler (1999). Note that some levels of model fit and factor loading strength were omitted to conserve space. TKL = Tucker, Koopman, and Linn; CB = Cudeck and Browne; WB = Wu and Browne.

Similar to CFI, the differences in TLI values resulting from the TKL_{CFI} and $\text{TKL}_{\text{RMSEA/CFI}}$ methods compared to the other model-error methods were most pronounced for conditions with many factors, weak factor loadings, and Poor model fit. This result (and the similar result for CFI) can be understood by thinking about how the relative and absolute fit indices differ in how they describe model fit. As an example, consider the an orthogonal model with ten major factors, five items per factor, and major common factor loadings fixed at .4. The population correlation matrix without model error for this condition ($\mathbf{\Omega}$) was 50×50 block-diagonal correlation matrix, with correlations of 0.16 between items that load on the same factor and zero otherwise. To obtain a population correlation matrix ($\mathbf{\Sigma}$) with sufficient model error to indicate poor model fit (based on an RMSEA value of 0.09), the elements of the $\mathbf{W}$ matrix had to be large enough to ensure that the square average squared difference between the off-diagonal elements of $\mathbf{\Omega}$ and $\mathbf{\Sigma}$ was 0.0081. Because the "noise" correlations from the minor common factors were relatively large compared to the non-zero elements of $\mathbf{\Omega}$, the $\mathbf{\Sigma}$ matrix did not have a clear factor structure. Thus, the major-factor model did not fit the $\mathbf{\Sigma}$ very well because the model was unable to account for the correlations between the items represented in the off-block-diagonal elements of $\mathbf{\Sigma}$ that were introduced by the $\mathbf{WW}'$ matrix. Moreover, because the $\mathbf{WW}'$ matrix increased the correlations between nearly all of the items that were uncorrelated in $\mathbf{\Omega}$, the independence model used as a baseline for computing CFI fit $\mathbf{\Sigma}$ relatively well.

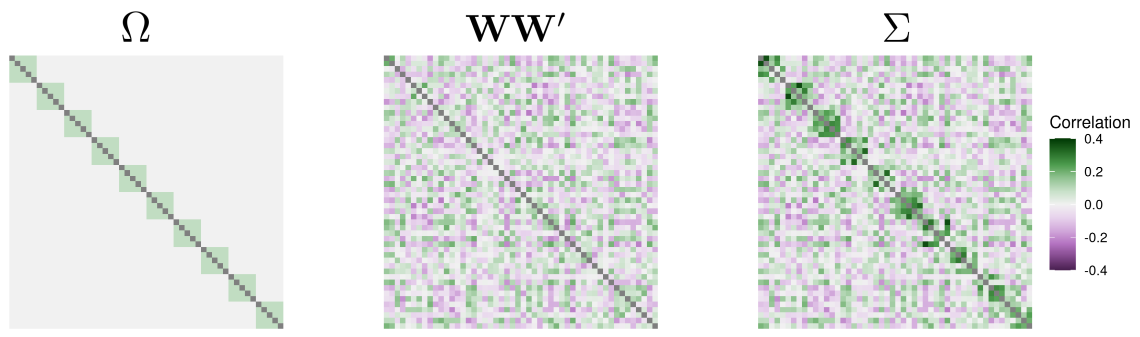

Figure 4.10. The population correlation matrix without model error ($\mathbf{\Sigma}$), the matrix of item correlations due to the minor common factors ($\mathbf{WW}'$), and the population matrix with model error ($\mathbf{\Sigma}$) for an orthogonal model with ten major factors, five items per factor, and major common factor loadings fixed at .4. The RMSEA and CFI values for this example were 0.09 and 0.25.

The effects of the model error introduced by the minor common factors can be seen in Figure Figure 4.10, which contains visual representations of $\mathbf{\Omega}$, WW$'$, and $\mathbf{\Sigma}$ matrices.

The $\mathbf{W}$ and $\mathbf{\Sigma}$ matrices shown in the figure were obtained using the TKL$_{\text{RMSEA}}$ method with a target RMSEA value of 0.09. The observed RMSEA and CFI values for $\mathbf{\Sigma}$ were 0.09 and 0.25. Notice in Figure Figure 4.10 how large the elements of the $\mathbf{WW}'$ matrix were relative to the $\mathbf{\Omega}$ matrix and how many off-block-diagonal elements there were relative to the block diagonal elements. This helps explain the incompatibility between RMSEA and CFI for models with many factors, weak major common factor loadings, and target RMSEA and CFI values reflecting poor model fit. To get an RMSEA value in line with the target value, all of the elements of $\mathbf{\Omega}$ were perturbed. The perturbations to the off-block-diagonal elements of $\mathbf{\Omega}$ were important in terms of the resulting CFI value because they were incompatible with the major common factor model (i.e., they should have been zero under that model) and therefore degraded the fit of the major factor model for $\mathbf{\Sigma}$ while improving the fit of the independence model. The effect of perturbing these off-block-diagonal elements became more pronounced as the number of factors increased because the number of off-block-diagonal elements increased faster than the number of block-diagonal elements as the total number of items increased. This can be clearly seen in Figure Figure 4.11, which shows the number of block diagonal elements and off-block-diagonal elements in the lower triangle of $\mathbf{\Sigma}$ as the number of factors increased for an orthogonal model with five items per factor.

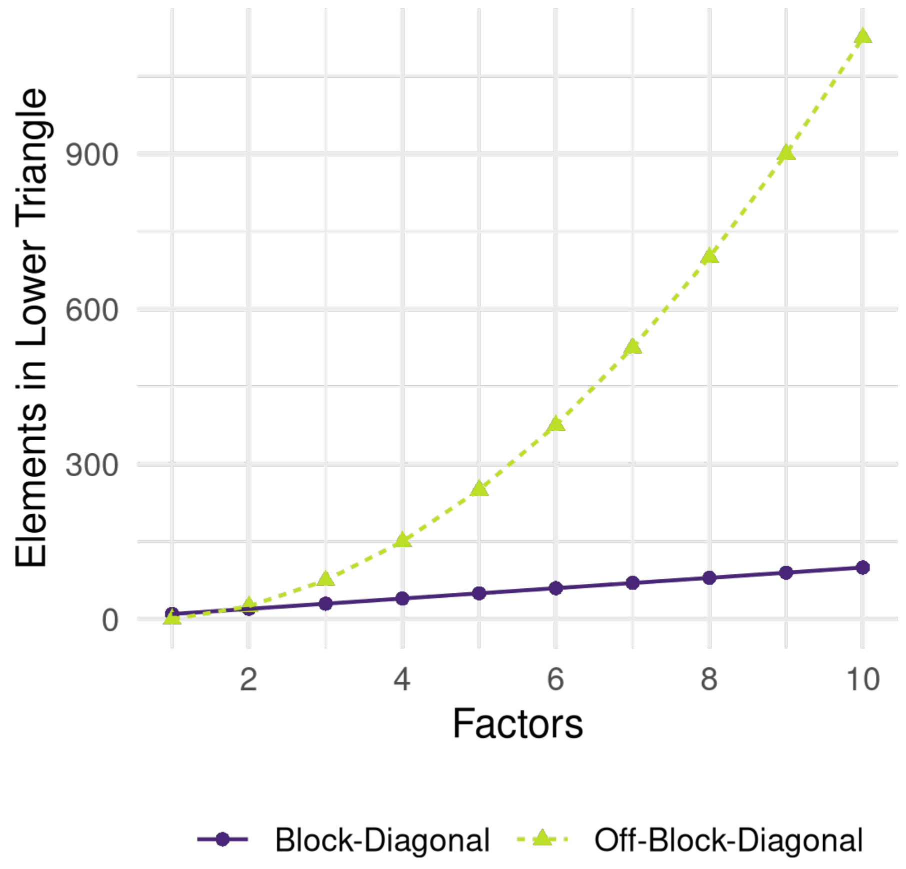

Figure 4.11. The number of block-diagonal elements in the lower-triangle of $\mathbf{\Sigma}$ compared to the number of off-block-diagonal elements for an orthogonal model with five items per factor.

4.4.4 CRMR

Just as the TLI results were similar to the CFI results, the CRMR results were similar in many ways to the RMSEA results. In particular,

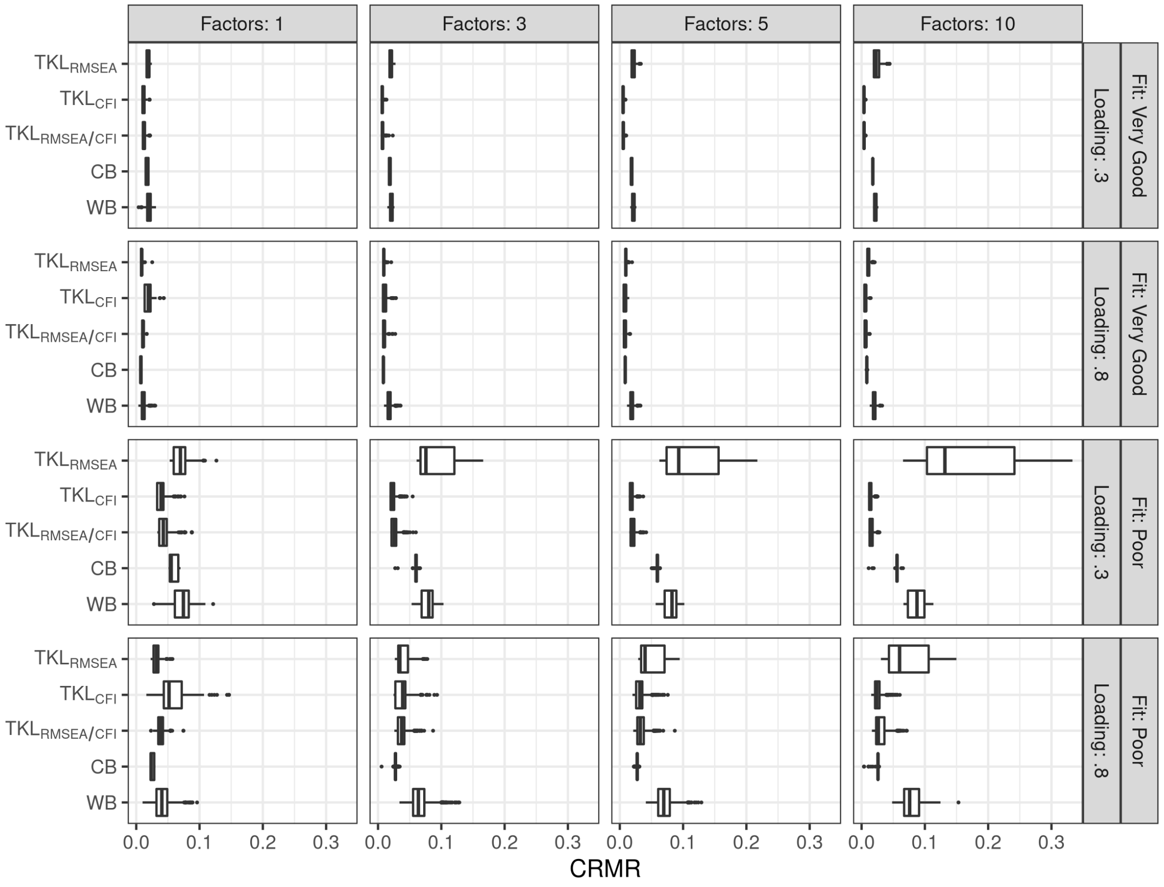

Figure 4.12. Distributions of the CRMR values for solutions produced by each of the model-error methods, conditioned on number of factors, model fit, and factor loading strength. The dashed lines indicate the threshold values of SRMR/CRMR that correspond to the targeted levels of model fit, according to Hu and Bentler (1999). Note that some levels of model fit and factor loading strength were omitted to conserve space. TKL = Tucker, Koopman, and Linn; CB = Cudeck and Browne; WB = Wu and Browne.

Chapter 5

Discussion

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas vel eros sed mauris porttitor semper nec a orci. Nullam vestibulum mi nec condimentum posuere. Pellentesque eget diam id sapien aliquet ullamcorper. Pellentesque blandit nec lectus ut mollis. Praesent in facilisis justo. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Sed eget congue leo, sed consequat libero. In rutrum malesuada nisi. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Morbi sollicitudin tortor ut sem facilisis mollis.

References

Beauducel, A., & Hilger, N. (2016). On the correlation of common factors with variance not accounted for by the factor model. *Communications in Statistics - Simulation and Computation*, *45*(6), 2145–2157. https://doi.org/gh64s7

Bentler, P. M. (1990). Comparative fit indexes in structural models. *Psychological Bulletin*, *107*(2), 238–246. https://doi.org/dbj

Bentler, P. M., & Bonett, D. G. (1980). Significance tests and goodness of fit in the analysis of covariance structures. *Psychological Bulletin*, *88*(3), 588–606. https://doi.org/dbm

Bezanson, J., Edelman, A., Karpinski, S., & Shah, V. B. (2017). Julia: A fresh approach to numerical computing. *SIAM Review*, *59*(1), 65–98. https://doi.org/f9wkpj

Bollen, K. A. (1989). A new incremental fit index for general structural equation models. *Sociological Methods & Research*, *17*(3), 303–316. https://doi.org/cfgdt5

Bollen, Kenneth A. (1989). *Structural Equations with Latent Variables*. John Wiley & Sons, Inc. https://doi.org/10.1002/9781118619179

Borges, J. L. (1998). On exactitude in science. In A. Hurley (Ed.), *Collected fictions*. New York:Viking.

Box, G. E. P., & Draper, N. R. (1987). *Empirical model-building and response surfaces.* (pp. xiv, 669). John Wiley & Sons.

Briggs, N. E., & MacCallum, R. C. (2003). Recovery of weak common factors by maximum likelihood and ordinary least squares estimation. *Multivariate Behavioral Research*, *38*(1), 25–56. https://doi.org/bgnz9w

Browne, M. W. (1984). Asymptotically distribution-free methods for the analysis of covariance structures. *British Journal of Mathematical and Statistical Psychology*, *37*(1), 62–83. https://doi.org/cd4vn7

Browne, M. W., & Cudeck, R. (1992). Alternative ways of assessing model fit. *Sociological Methods & Research*, *21*(2), 230–258. https://doi.org/dbn

Browne, M. W., MacCallum, R. C., Kim, C. T., Andersen, B. L., & Glaser, R. (2002). When fit indices and residuals are incompatible. *Psychological Methods*, *7*(4), 403–421. https://doi.org/fhq5t3

Browne, M. W., & Shapiro, A. (1988). Robustness of normal theory methods in the analysis of linear latent variate models. *British Journal of Mathematical and Statistical Psychology*, *41*(2), 193–208. https://doi.org/bdk8xk

Byrd, R. H., Lu, P., Nocedal, J., & Zhu, C. (1995). A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, *16*(5), 1190–1208. https://doi.org/bpjm24

Carroll, L. (1894). *Sylvie and Bruno, Volume II, by Lewis Carroll, illustrated by Harry Furniss*. https://www.gutenberg.org/files/48795/48795-h/48795-h.htm.

Chung, S., & Cai, L. (2019). Alternative multiple imputation inference for categorical structural equation modeling. *Multivariate Behavioral Research*, *54*(3), 323–337. https://doi.org/gh64rk

Cudeck, R. (1989). Analysis of correlation matrices using covariance structure models. *Psychological Bulletin*, *105*(2), 317–327. https://doi.org/fqks6n

Cudeck, R., & Browne, M. W. (1992). Constructing a covariance matrix that yields a specified minimizer and a specified minimum discrepancy function value. *Psychometrika*, *57*(3), 357–369. https://doi.org/cq6ckd

Cudeck, R., & Henly, S. J. (1991). Model selection in covariance structures analysis and the "problem" of sample size: A clarification. *Psychological Bulletin*, *109*(3), 512. https://doi.org/bpqvdv

de Winter, J. C. F., & Dodou, D. (2016). Common factor analysis versus principal component analysis: A comparison of loadings by means of simulations. *Communications in Statistics - Simulation and Computation*, *45*(1), 299–321. https://doi.org/gh64cn

Eco, U. (1994). On the impossibility of drawing a map of the empire on a scale of 1 to 1. In W. Weaver & U. Eco (Eds.), *How to travel with a salmon & other essays* (1st ed..). New York:Harcourt, Brace.

Gelman, A. (2008). Some thoughts on the saying, "All models are wrong, but some are useful." In *Statistical Modeling, Causal Inference, and Social Science*. https://statmodeling.stat.columbia

Gnambs, T., & Staufenbiel, T. (2016). Parameter accuracy in meta-analyses of factor structures. *Research Synthesis Methods*, *7*(2), 168–186. https://doi.org/gcz6x6

Gupta, A. K., & Nagar, D. K. (2000). *Matrix variate distributions*. Chapman & Hall.

Hair, J. F., Black, W. C., Babin, B. J., & Anderson, R. E. (2018). *Multivariate Data Analysis*. Cengage.

Holland, J. H. (1975). *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*. University of Michigan Press.

Hong, S. (1999). Generating correlation matrices with model error for simulation studies in factor analysis: A combination of the Tucker-Koopman-Linn model and Wijsman's algorithm. *Behavior Research Methods, Instruments, & Computers*, *31*(4), 727–730. https://doi.org/dsjcnm

Howe, W. G. (1955). *Some contributions to factor analysis* (Technical {{Report}} ONRL-1919).

Hsu, H.-Y., Kwok, O., Lin, J. H., & Acosta, S. (2015). Detecting misspecified multilevel structural equation models with common fit indices: A Monte Carlo study. *Multivariate Behavioral Research*, *50*(2), 197–215. https://doi.org/gg5fk7

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

Jackson, D. L. (2009). Reporting practices in confirmatory factor analysis: An overview and some recommendations. *Psychological Methods*, *14*(1), 6. https://doi.org/cbd6zd

Jöreskog, K. G. (1969). A general approach to confirmatory maximum likelihood factor analysis. *Psychometrika*, *34*(2, Pt.1), 183–202. https://doi.org/cs6df7

Kline, R. B. (2011). *Principles and practice of structural equation modeling* (3rd ed). Guilford Press.

Kracht, J. D., & Waller, N. G. (2020). Assessing dimensionality in non-positive definite tetrachoric correlation matrices: Does matrix smoothing help? *Multivariate Behavioral Research*. https://doi.org/ghq7p7

Lai, K. (2020a). Better confidence intervals for RMSEA in growth models given nonnormal data. *Structural Equation Modeling: A Multidisciplinary Journal*, *27*(2), 255–274. https://doi.org/gh7r8k

Lai, K. (2018). Estimating standardized SEM parameters given nonnormal data and incorrect model: Methods and comparison. *Structural Equation Modeling: A Multidisciplinary Journal*, *25*(4), 600–620. https://doi.org/gc4pjh

Lai, K. (2020b). Correct estimation methods for RMSEA under missing data. *Structural Equation Modeling: A Multidisciplinary Journal*, *0*(0), 1–12. https://doi.org/gh7r9d

Lai, K. (2020c). Fit difference between nonnested models given categorical data: Measures and estimation. *Structural Equation Modeling: A Multidisciplinary Journal*, *0*(0), 1–22. https://doi.org/gh7r9f

Lai, K. (2019). A simple analytic confidence interval for CFI given nonnormal data. *Structural Equation Modeling: A Multidisciplinary Journal*, *26*(5), 757–777. https://doi.org/gh7r8d

Lai, K. (2020d). Correct point estimator and confidence interval for RMSEA given categorical data. *Structural Equation Modeling: A Multidisciplinary Journal*, *27*(5), 678–695. https://doi.org/gh7r82

Lai, K., & Green, S. B. (2016). The problem with having two watches: Assessment of fit when RMSEA and CFI disagree. *Multivariate Behavioral Research*, *51*(2-3), 220–239. https://doi.org/10.1080/00273171.2015.1134306

Lai, K., & Zhang, X. (2017). Standardized parameters in misspecified structural equation models: Empirical performance in point estimates, standard errors, and confidence intervals. *Structural Equation Modeling: A Multidisciplinary Journal*, *24*(4), 571–584.

https://doi.org/gcph63

Lorenzo-Seva, U., & Ferrando, P. J. (2020). Unrestricted factor analysis of multidimensional test items based on an objectively refined target matrix. *Behavior Research Methods*, *52*(1), 116–130. https://doi.org/gh64j4

Lorenzo-Seva, U., & Ginkel, J. R. V. (2016). Multiple imputation of missing values in exploratory factor analysis of multidimensional scales: Estimating latent trait scores. *Anales de Psicología / Annals of Psychology*, *32*(2), 596–608. https://doi.org/dvrt

MacCallum, R. C., & Tucker, L. R. (1991). Representing sources of error in the common-factor model: Implications for theory and practice. *Psychological Bulletin*, *109*(3), 502–511. https://doi.org/cgsqhv

MacCallum, R. C., Widaman, K. F., Preacher, K. J., & Hong, S. (2001). Sample size in factor analysis: The role of model error. *Multivariate Behavioral Research*, *36*(4), 611–637. https://doi.org/b3v4p6

Marsh, H. W., Hau, K.-T., & Grayson, D. (2005). Goodness of fit in structural equation models. In *Contemporary psychometrics: A festschrift for Roderick P. McDonald.* (pp. 275–340). Lawrence Erlbaum Associates Publishers.

Maydeu-Olivares, A. (2017). Assessing the size of model misfit in structural equation models. *Psychometrika*, *82*(3), 533–558. https://doi.org/gbthc8

Meehl, P. E., & Waller, N. G. (2002). The path analysis controversy: A new statistical approach to strong appraisal of verisimilitude. *Psychological Methods*, *7*(3), 283–300. https://doi.org/dbzsmd

Miles, J., & Shevlin, M. (2007). A time and a place for incremental fit indices. *Personality and Individual Differences*, *42*(5), 869–874. https://doi.org/fhdt3m

Mitchell, M. (1996). *An introduction to genetic algorithms* (I. NetLibrary, Ed.). MIT Press.

Montoya, A. K., & Edwards, M. C. (2020). The poor fit of model fit for selecting number of factors in exploratory factor analysis for scale evaluation. *Educational and Psychological Measurement*, 0013164420942899. https://doi.org/gh7r86

Mulaik, S. A. (2009). *Foundations of factor analysis*. Chapman and Hall/CRC. https://doi.org/10.1201/b15851

Myers, N. D., Jin, Y., Ahn, S., Celimli, S., & Zopluoglu, C. (2015). Rotation to a partially specified target matrix in exploratory factor analysis in practice. *Behavior Research Methods*, *47*(2), 494–505. https://doi.org/f7jmdz

Nester, M. R. (1996). An applied statistician's creed. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, *45*(4), 401–410. https://doi.org/bk59hb

Neuman, G. A., Bolin, A. U., & Briggs, T. E. (2000). Identifying General Factors of Intelligence: A Confirmatory Factor Analysis of the Ball Aptitude Battery. *Educational and Psychological Measurement*, *60*(5), 697–712. https://doi.org/dxx8x4

Ogasawara, H. (2001). Approximations to the distributions of fit indexes for misspecified structural equation models. *Structural Equation Modeling: A Multidisciplinary Journal*, *8*(4), 556–574. https://doi.org/bv4mwk

Olsson, U. H., Foss, T., & Breivik, E. (2004). Two equivalent discrepancy functions for maximum likelihood estimation: Do their test statistics follow a non-central chi-square distribution under model misspecification? *Sociological Methods & Research*, *32*(4), 453–500. https://doi.org/ft43gk

Pavlov, G., Maydeu-Olivares, A., & Shi, D. (2021). Using the Standardized Root Mean Squared Residual (SRMR) to Assess Exact Fit in Structural Equation Models. *Edu-*

cational and Psychological Measurement, *81*(1), 110–130. https://doi.org/gmchg6

Pek, J. (2012). *Fungible parameter contours and confidence regions in structural equation models* [PhD thesis, University of North Carolina]. https://doi.org/10.17615/S9V9-5M31

Porritt, M. (2015). *Performance of number of factors procedures in small sample sizes* [PhD thesis]. Loma Linda University.

Preacher, K. J., & MacCallum, R. C. (2002). Exploratory factor analysis in behavior genetics research: Factor recovery with small sample sizes. *Behavior Genetics*, *32*(2), 153–161. https://doi.org/cghpnv

Preacher, K. J., Zhang, G., Kim, C., & Mels, G. (2013). Choosing the optimal number of factors in exploratory factor analysis: A model selection perspective. *Multivariate Behavioral Research*, *48*(1), 28–56. https://doi.org/gckf6t

R Core Team. (2021). *R: A language and environment for statistical computing* [Manual]. R Foundation for Statistical Computing.

Satorra, A. (2015). A Comment on a paper by H. Wu and M. W. Browne (2014). *Psychometrika*, *80*(3), 613–618. https://doi.org/gjrkc3

Scrucca, L. (2013). GA: A package for genetic algorithms in R. *Journal of Statistical Software*, *53*(1), 1–37. https://doi.org/gft29t

Shapiro, A. (1983). Asymptotic distribution theory in the analysis of covariance structures. *South African Statistical Journal*, *17*(1), 33–81. https://doi.org/10.10520/AJA0038271X_800

Shapiro, A. (2007). Statistical inference of moment structures. In *Handbook of latent variable and related models* (1st ed.., pp. 229–260). Amsterdam.

Steele, J. M. (2008). *Models: Masterpieces and lame excuses.* http://www-stat.wharton.upenn.edu/~steele

Steiger, J. H. (1990). Structural model evaluation and modification: An interval estimation approach. *Multivariate Behavioral Research*, *25*(2), 173–180. https://doi.org/db5

Steiger, J. H. (2007). Understanding the limitations of global fit assessment in structural equation modeling. *Personality and Individual Differences*, *42*(5), 893–898. https://doi.org/dsxstk

Steiger, J. H. (1989). *EzPATH: Causal modeling: A supplementary module for SYSTAT and SYGRAPH: PC-MS-DOS, version 1.0*. Systat.

The Mathworks, Inc. (2019). *MATLAB and statistics toolbox R2019a*.

Tomarken, A. J., & Waller, N. G. (2003). Potential problems with "well fitting" models. *Journal of Abnormal Psychology*, *112*(4), 578. https://doi.org/fpmxjq

Trichtinger, L. A., & Zhang, G. (2020). Quantifying model error in P-technique factor analysis. *Multivariate Behavioral Research*, *0*(0), 1–16. https://doi.org/gh64s8

Tucker, L. R., Koopman, R. F., & Linn, R. L. (1967). *Evaluation of factor analytic research procedures by means of simulated correlation matrices*. Department of Psychology, University of Illinois, Urbana.

Tucker, L. R., Koopman, R. F., & Linn, R. L. (1969). Evaluation of factor analytic research procedures by means of simulated correlation matrices. *Psychometrika*, *34*(4), 421–459. https://doi.org/chcxvf

Tucker, L. R., & Lewis, C. (1973). A reliability coefficient for maximum likelihood factor analysis. *Psychometrika*, *38*(1), 1–10. https://doi.org/bcz7k9

Vaart, A. W. van der. (1998). *Asymptotic Statistics* (First). Cambridge University Press. https://doi.org/10.1017/CBO9780511802256

Waller, N. G. (2021). *Fungible: Psychometric functions from the Waller lab.* [Manual].

Wu, H., & Browne, M. W. (2015a). Quantifying adventitious error in a covariance structure as a random effect. *Psychometrika*, *80*(3), 571–600. https://doi.org/gjrkc4

Wu, H., & Browne, M. W. (2015b). Random model discrepancy: Interpretations and technicalities (a rejoinder). *Psychometrika*, *80*(3), 619–624. https://doi.org/ggfngm

Xia, Y. (2021). Determining the number of factors when population models can be closely approximated by parsimonious models. *Educational and Psychological Measurement*, 0013164421992836. https://doi.org/gh68s2

Xia, Y., & Yang, Y. (2019). RMSEA, CFI, and TLI in structural equation modeling with ordered categorical data: The story they tell depends on the estimation methods. *Behavior Research Methods*, *51*(1), 409–428. https://doi.org/ggssdx

Xia, Y., Yung, Y.-F., & Zhang, W. (2016). Evaluating the selection of normal-theory weight matrices in the Satorra–Bentler correction of chi-square and standard errors. *Structural Equation Modeling: A Multidisciplinary Journal*, *23*(4), 585–594. https://doi.org/gcz6zq

Yuan, K.-H., & Marshall, L. L. (2004). A new measure of misfit for covariance structure models. *Behaviormetrika*, *31*(1), 67–90. https://doi.org/cr2mb8

Appendix A

R Code

A.1 Implementations of Model Error Methods

The implementations of the model error methods used are also available bundled as an R package, *noisemaker*, which can be downloaded from https://www.github.com/JustinKracht/noisemaker.

```r
#' Cudeck & Browne (1992) model error method
#'
#' Generate a population correlation matrix using the model described in Cudeck
#' and Browne (1992).
#'
#' @param mod A `fungible::simFA()` model object.
#' @param target_rmsea (scalar) Target RMSEA value.
#' @export
#' @references Cudeck, R., & Browne, M. W. (1992). Constructing a covariance
#'   matrix that yields a specified minimizer and a specified minimum
#'   discrepancy function value. *Psychometrika*, *57*(3), 357-369.
#'   <https://doi.org/10/cq6ckd>

cb <- function(mod,
               target_rmsea) {

  if (target_rmsea < 0 | target_rmsea > 1) {
    stop("The target RMSEA value must be a number between 0 and 1.\n",
         crayon::cyan("\u2139"), " You've specified a target RMSEA value of ",
         target_rmsea, ".", call. = F)
  }

  if (!(is.list(mod)) |
      is.null(mod$loadings) |
      is.null(mod$Phi) |
      is.null(mod$Rpop)) {
    stop("`mod` must be a valid `simFA()` model object.", call. = F)
  }

  p <- nrow(mod$loadings)
  k <- ncol(mod$loadings)
  df <- (p * (p - 1) / 2) - (p * k) + (k * (k - 1) / 2)
  discrep <- target_rmsea^2 * df
  sem_mod <- semify(mod)

  Sigma <- MBESS::Sigma.2.SigmaStar(
    model = sem_mod$model,
    model.par = sem_mod$theta,
    latent.var = sem_mod$latent_var,
    discrep = discrep
  )$Sigma.star

  # Check positive definiteness
  lambda_min <- min(eigen(Sigma, symmetric = TRUE, only.values = TRUE)$values)
  if (lambda_min < 0) {
    stop("Sigma is indefinite.\n",
         crayon::cyan("\u2139"), " The minimum eigenvalue is ",
         round(lambda_min, 2), call. = FALSE)
  }

  Sigma
}
#' Calculate CFI for two correlation matrices
#'
#' Given two correlation matrices of the same dimension, calculate the CFI value
#' value using the independence model as the null model.
#'
#' @param Sigma (matrix) Population correlation or covariance matrix (with model
#'   error).
#' @param Omega (matrix) Model-implied population correlation or covariance
#'   matrix.
#'
#' @export
#'
#' @examples
#' library(fungible)
#' library(noisemaker)
#'
#' mod <- fungible::simFA(Model = list(NFac = 3),
#'                        Seed = 42)
#' set.seed(42)
#' Omega <- mod$Rpop
#' Sigma <- noisemaker(
#'   mod = mod,
#'   method = "CB",
#'   target_rmsea = 0.05
#' )$Sigma
#' cfi(Sigma, Omega)
cfi <- function(Sigma, Omega) {
  if (!is.matrix(Sigma) | !is.matrix(Omega)) {
    stop("Sigma and Omega must be matrices.", call. = F)
  } else if (all.equal(Sigma, t(Sigma)) != TRUE) {
    stop("Sigma must be a symmetric matrix.", call. = F)
  } else if (all.equal(Omega, t(Omega)) != TRUE) {
    stop("Omega must be a symmetric matrix.", call. = F)
  } else if (all.equal(dim(Omega), dim(Sigma)) != TRUE) {
    stop("Sigma and Omega must have the same dimensions.", call. = F)
  }

  p <- nrow(Sigma)
  Ft <- log(det(Omega)) - log(det(Sigma)) +
    sum(diag(Sigma %*% solve(Omega))) - p
  cfi <- 1 - (Ft / -log(det(Sigma)))
  cfi
}
#' Find an `lm` model to use with the Wu & Browne (2015) model error method
#'
#' The Wu & Browne (2015) model error method takes advantage of the relationship
#' between v and RMSEA:
#'
#' \deqn{v = RMSEA^2 + o(RMSEA^2).}
#'
#' As RMSEA increases, the approximation \eqn{v ~= RMSEA^2} becomes worse. This
#' function generates population correlation matrices with model error for
#' multiple target RMSEA values and then regresses the target RMSEA values on
#' the median observed RMSEA values for each target. The fitted model can then
#' be used to predict a `target_rmsea` value that will give solutions with RMSEA
#' values that are close to the desired value.
#'
#' @param mod A `fungible::simFA()` model object.
#' @param n The number of times to evaluate `wb()` at each point.
#' @param values The number of target RMSEA values to evaluate between 0.02 and
#'   0.1.
#' @param lower (scalar) The smallest target RMSEA value to use.
#' @param upper (scalar) The largest target RMSEA value to use.
#'
#' @return (`lm` object) An `lm` object to use with the \code{\link{wb}}
#'   function to obtain population correlation matrices with model error that
#'   have RMSEA values closer to the target RMSEA values. The `lm` object will
#'   predict a `target_rmsea` value that will give solutions with (median) RMSEA
#'   values close to the desired RMSEA value.
```

```r
#' @export
#'
#' @examples
#' mod <- fungible::simFA(Seed = 42)
#' set.seed(42)
#' wb_mod <- get_wb_mod(mod)
#' noisemaker(mod, method = "WB", target_rmsea = 0.05, wb_mod = wb_mod)

get_wb_mod <- function(mod, n = 50, values = 10, lower = .01, upper = .095) {
  # Check arguments
  if (!(is.list(mod))) {
    stop("`mod` must be a valid `simFA()` model object.", call. = F)
  }
  if (is.null(mod$loadings) |
      is.null(mod$Phi) |
      is.null(mod$Rpop)) {
    stop("`mod` must be a valid `simFA()` model object.", call. = F)
  }
  if (length(n) != 1L | !is.numeric(n) | n <= 0) {
    stop("`n` must be a number greater than zero.\n",
         crayon::cyan("\u2139"), " You've specified an `n` value of ",
         n, ".", call. = F)
  }
  if (length(values) != 1L | !is.numeric(values) | values < 2) {
    stop("`values` must be a number greater than two.\n",
         crayon::cyan("\u2139"), " You've specified a `values` value of ",
         values, ".", call. = F)
  }
  if (length(lower) != 1L | !is.numeric(lower) | lower <= 0) {
    stop("`lower` must be a number greater than zero.\n",
         crayon::cyan("\u2139"), " You've specified a `lower` value of ",
         lower, ".", call. = F)
  }
  if (length(upper) != 1L | !is.numeric(upper)) {
    stop("`upper` must be a number.\n",
         crayon::cyan("\u2139"), " You've specified an `upper` value of ",
         upper, ".", call. = F)
  }

  k <- ncol(mod$loadings)
  Omega <- mod$Rpop
  p <- nrow(Omega)

  # WB requires m < p; calculate upper bound
  # (1 / target_rmsea^2) > p means that target_rmsea < 1 / sqrt(p)
  max_target_rmsea <- 1 / sqrt(p)
  if (upper >= max_target_rmsea) {
    warning("Specified upper bound was too large and was reduced.")
    # Set new upper bound to the maximum target RMSEA, minus 5% to avoid hitting
    # a boundary
    upper <- max_target_rmsea - 0.05 * max_target_rmsea
  }

  rmsea_values <- seq(lower, upper, length.out = values)

  rmsea_medians <- sapply(
    X = rmsea_values,
    FUN = function(target_rmsea,
                   mod,
                   Omega,
                   k) {
      obs_rmsea <- replicate(n = n, expr = {
        rmsea(wb(mod, target_rmsea, adjust_target = FALSE)$Sigma, Omega, k)
      })
      stats::median(obs_rmsea)
    },
    mod = mod,
    Omega = Omega,
    k = k
  )

  m1 <- stats::lm(rmsea_values ~ poly(rmsea_medians, 2))
  m1
}
#' Simulate a population correlation matrix with model error
#'
#' This tool lets the user generate a population correlation matrix with model
#' error using one of three methods: (1) the Tucker, Koopman, and Linn (TKL;
#' 1969) method, (2) the Cudeck and Browne (CB; 1992) method, or (3) the Wu and
#' Browne (WB; 2015) method. If the CB or WB methods are used, the user can
#' specify the desired RMSEA value. If the TKL method is used, an optimization
#' procedure finds a solution that produces RMSEA and/or CFI values that are
#' close to the user-specified values.
#'
#' @param mod A `fungible::simFA()` model object.
#' @param method (character) Model error method to use ("TKL", "CB", or "WB").
#' @param target_rmsea (scalar) Target RMSEA value.
#' @param target_cfi (scalar) Target CFI value.
#' @param tkl_ctrl (list) A control list containing the following TKL-specific
#'   arguments. See the `tkl()` help file for more details.
#' @param wb_mod (`lm` object) An optional `lm` object used to find a target
#'   RMSEA value that results in solutions with RMSEA values close to the
#'   desired value. Note that if no `wb_mod` is provided, a model will be
#'   estimated at run time. If many population correlation matrices are going to
#'   be simulated using the same model, it will be considerably faster to
#'   estimate `wb_mod` ahead of time. See also `get_wb_mod()`.
```

```
#'
#' @return A list containing \eqn{\Sigma}, RMSEA and CFI values, and the TKL
#'   parameters (if applicable).
#' @export
#'
#' @examples
#' mod <- fungible::simFA(Seed = 42)
#'
#' set.seed(42)
#' # Simulate a population correlation matrix using the TKL method with target
#' # RMSEA and CFI values specified.
#' noisemaker(mod, method = "TKL",
#'            target_rmsea = 0.05,
#'            target_cfi = 0.95,
#'            tkl_ctrl = list(optim_type = "optim"))
#'
#' # Simulate a population correlation matrix using the CB method with target
#' # RMSEA value specified.
#' noisemaker(mod, method = "CB",
#'            target_rmsea = 0.05)
#'
#' # Simulation a population correlation matrix using the WB method with target
#' # RMSEA value specified.
#' noisemaker(mod,
#'            method = "WB",
#'            target_rmsea = 0.05)
noisemaker <- function(mod,
                       method = c("TKL", "CB", "WB"),
                       target_rmsea = 0.05,
                       target_cfi = NULL,
                       tkl_ctrl = list(),
                       wb_mod = NULL) {

  if (is.null(target_rmsea) & is.null(target_cfi)) {
    stop("Either target RMSEA or target CFI must be specified.",
         call. = F)
  }
  if (!is.numeric(target_rmsea) & !is.null(target_rmsea)) {
    stop("Target RMSEA value must be a number or NULL.\n",
         crayon::cyan("\u2139"), " You've specified a target RMSEA value of ",
         target_rmsea, ".", call. = F)
  }
  if (!is.numeric(target_cfi) & !is.null(target_cfi)) {
    stop("Target CFI value must be either a number or NULL.\n",
         crayon::cyan("\u2139"), " You've specified a target CFI value of ",
         target_cfi, ".", call. = F)
  }
  if (!is.null(target_rmsea)) {
    if (target_rmsea < 0 | target_rmsea > 1) {
      stop("The target RMSEA value must be a number between 0 and 1.\n",
           crayon::cyan("\u2139"), " You've specified a target RMSEA value of ",
           target_rmsea, ".", call. = F)
    }
  }
  if (!is.null(target_cfi) & (method != "TKL")) {
    stop(
      "The TKL method must be used when a CFI value is specified.\n",
      crayon::cyan("\u2139")," You've selected the ", method," method.\n",
      crayon::cyan("\u2139")," You've specified a target CFI value of ",
      target_cfi, "."
    )
  }
  if (!(method %in% c("TKL", "WB", "CB"))) {
    stop("`method` must be `TKL`, `CB`, or `WB`.\n",
         crayon::cyan("\u2139"), " You've specified ",
         method, " as `method`.", call. = F)
  }
  if (!(is.list(mod)) |
      is.null(mod$loadings) |
      is.null(mod$Phi) |
      is.null(mod$Rpop)) {
    stop("`mod` must be a valid `simFA()` model object.", call. = F)
  }
  if (mod$cn$ModelError$ModelError == TRUE) {
    warning(paste0("The `simFA()` object you provided includes model error",
                   " parameters that will be ignored by this function."))
  }

  out_list <- list(Sigma = NA,
                   rmsea = NA,
                   cfi = NA,
                   fn_value = NA,
                   m = NA,
                   v = NA,
                   eps = NA,
                   W = NA)

  k <- ncol(mod$loadings) # number of major factors

  if (method == "WB") {
    wb_out <- wb(mod = mod,
                 target_rmsea = target_rmsea,
                 wb_mod = wb_mod)
    out_list$Sigma <- wb_out$Sigma
    out_list$rmsea <- rmsea(out_list$Sigma, mod$Rpop, k)
    out_list$cfi <- cfi(out_list$Sigma, mod$Rpop)
```

```r
    out_list$m <- wb_out$m
  } else if (method == "CB") {
    out_list$Sigma <- cb(mod = mod,
                         target_rmsea = target_rmsea)
    out_list$rmsea <- rmsea(out_list$Sigma, mod$Rpop, k)
    out_list$cfi <- cfi(out_list$Sigma, mod$Rpop)
  } else if (method == "TKL") {
    tkl_out <- tkl(mod = mod,
                   target_rmsea = target_rmsea,
                   target_cfi = target_cfi,
                   tkl_ctrl = tkl_ctrl)

    out_list$Sigma <- tkl_out$RpopME
    out_list$rmsea <- tkl_out$rmsea
    out_list$cfi <- tkl_out$cfi
    out_list$v <- tkl_out$v
    out_list$eps <- tkl_out$eps
    out_list$W <- tkl_out$W
    out_list$fn_value <- tkl_out$fn_value
  }

  out_list
}
#' Objective function for optimizing RMSEA and CFI
#'
#' This is the objective function that is minimized by the `tkl()` function.
#'
#' @param par (vector) Values of model error variance (\eqn{\upsilon}) and
#'   epsilon (\eqn{\epsilon}).
#' @param Rpop (matrix) The model-implied correlation matrix.
#' @param W (matrix) Matrix of provisional minor common factor loadings with
#'   unit column variances.
#' @param p (scalar) Number of variables.
#' @param u (vector) Major common factor variances.
#' @param df (scalar) Model degrees of freedom.
#' @param target_rmsea (scalar) Target RMSEA value.
#' @param target_cfi (scalar) Target CFI value.
#' @param weights (vector) Vector of length two indicating how much weight to
#'   give RMSEA and CFI, e.g., `c(1,1)` (default) gives equal weight to both
#'   indices; `c(1,0)` ignores the CFI value.
#' @param WmaxLoading (scalar) Threshold value for `NWmaxLoading`.
#' @param NWmaxLoading (scalar) Maximum number of absolute loadings \eqn{\ge}
#'   `WmaxLoading` in any column of `W`.
#' @param penalty (scalar) Large (positive) penalty value to apply if the
#'   NWmaxLoading condition is violated.
#' @param return_values (boolean) If `TRUE`, return the objective function value
#'   along with `Rpop`, `RpopME`, `W`, `RMSEA`, `CFI`, `v`, and `eps` values. If
#'   `FALSE`, return only the objective function value.
```

```r
#'
#' @export

obj_func <- function(par = c(v, eps),
                     Rpop, W, p, u, df,
                     target_rmsea, target_cfi,
                     weights = c(1, 1),
                     WmaxLoading = NULL,
                     NWmaxLoading = 2,
                     penalty = 0,
                     return_values = FALSE) {
  v <- par[1] # error variance
  eps <- par[2] # epsTKL

  # Rescale W using eps
  scaling_matrix <- diag((1 - eps)^(0:(ncol(W) - 1)))
  W <- W %*% scaling_matrix

  # Create W matrix such that the proportion of unique variance accounted for by
  # the minor common factors is v.
  # Adapted from simFA() (lines 691--698)
  wsq <- diag(tcrossprod(W))
  ModelErrorVar <- v * u
  W <- diag(sqrt(ModelErrorVar / wsq)) %*% W
  RpopME <- Rpop + tcrossprod(W)
  diag(RpopME) <- 1

  # ML objective function value for the full model
  # Adapted from simFA() (lines 651--660)
  Ft <- log(det(Rpop)) - log(det(RpopME)) +
    sum(diag(RpopME %*% solve(Rpop))) - p

  # ML objective function value for the baseline (independence) model
  Fb <- -log(det(RpopME))

  # Compute RMSEA and CFI values
  # Adapted from simFA() (lines 651--660)
  rmsea <- sqrt(Ft / df)
  cfi <- 1 - (Ft / -log(det(RpopME)))

  # Define penalty if WmaxLoading and NWmaxLoading are defined
  if (!is.null(WmaxLoading)) {
    # Takes the value 1 if any column of W has more than NWmaxLoading
    # abs(loadings) >= WmaxLoading
    max_loading_indicator <- any(
      max(apply(abs(W) >= WmaxLoading, 2, sum)) > NWmaxLoading
    )
  } else {
    max_loading_indicator <- 0
  }

  weights <- weights / sum(weights) # scale weights to sum to one

  # Compute objective function value
  # fn_value <- weights[1] * (rmsea - target_rmsea)^2 +
  #   weights[2] * (cfi - target_cfi)^2 +
  #   penalty * max_loading_indicator
  fn_value <- weights[1] * ((rmsea - target_rmsea)^2 / target_rmsea^2) +
    weights[2] * (((1 - cfi) - (1 - target_cfi))^2 / (1 - target_cfi)^2) +
    penalty * max_loading_indicator

  # Objective function value weights RMSEA and CFI differences equally; could be
  # changed, if necessary
  if (return_values == FALSE) {
    fn_value
  } else {
    names(fn_value) <- names(v) <- names(eps) <- NULL
    list(
      fn_value = fn_value,
      Rpop = Rpop,
      RpopME = RpopME,
      W = W,
      rmsea = rmsea,
      cfi = cfi,
      v = v,
      eps = eps
    )
  }
}
#' Calculate RMSEA between two correlation matrices
#'
#' Given two correlation matrices of the same dimension, calculate the RMSEA
#' value using the degrees of freedom for the exploratory factor analysis model
#' (see details).
#'
#' @param Sigma (matrix) Population correlation or covariance matrix (with model
#'   error).
#' @param Omega (matrix) Model-implied population correlation or covariance
#'   matrix.
#' @param k (scalar) Number of major common factors.
#'
#' @details Note that this function uses the degrees of freedom for an
#'   exploratory factor analysis model: \deqn{df = p(p-1)/2-(pk)+k(k-1)/2,}
#'   where \eqn{p} is the number of items and \eqn{k} is the number of major
#'   factors.
#'
```

```r
#' @md
#' @export
#'
#' @examples
#' mod <- fungible::simFA(Model = list(NFac = 3),
#'                        Seed = 42)
#' set.seed(42)
#' Omega <- mod$Rpop
#' Sigma <- noisemaker(
#'   mod = mod,
#'   method = "CB",
#'   target_rmsea = 0.05
#' )$Sigma
#' rmsea(Sigma, Omega, k = 3)

rmsea <- function(Sigma, Omega, k) {
  if (!is.matrix(Sigma) | !is.matrix(Omega)) {
    stop("Sigma and Omega must be matrices.", call. = F)
  } else if (all.equal(Sigma, t(Sigma)) != TRUE) {
    stop("Sigma must be a symmetric matrix.", call. = F)
  } else if (all.equal(Omega, t(Omega)) != TRUE) {
    stop("Omega must be a symmetric matrix.", call. = F)
  } else if (all.equal(dim(Omega), dim(Sigma)) != TRUE) {
    stop("Sigma and Omega must have the same dimensions.", call. = F)
  } else if (!is.numeric(k) | ((k %% 1) != 0) | k < 0) {
    stop("`k` must be a non-negative integer.\n",
         crayon::cyan("\u2139"), " You've specified ", k, " as `k`.", call. = F)
  }

  p <- nrow(Sigma)
  df <- (p * (p - 1) / 2) - (p * k) + (k * (k - 1) / 2)
  Fm <- log(det(Omega)) - log(det(Sigma)) + sum(diag(Sigma %*% solve(Omega))) - p
  sqrt(Fm / df)
}
#' Generate an sem model from a simFA model object
#'
#' @param mod A `fungible::simFA()` model object.
#'
#' @export
#'
#' @examples
#' ex_mod <- fungible::simFA(Seed = 42)
#' semify(mod = ex_mod)
semify <- function(mod) {
  L <- mod$loadings
  Phi <- mod$Phi
  u <- 1 - mod$h2
```

```r
# Specify factor loadings
loading_spec <- ""
loadings <- numeric(length = sum(L != 0))
loading_names <- character(length = sum(L != 0))
i <- 1
for (item in seq_len(nrow(L))) {
  for (factor in seq_len(ncol(L))) {
    if (L[item, factor] == 0) next
    loading_spec <- paste0(
      loading_spec, "F", factor, " -> ", "V", item,
      ", lambda", i, ", ", L[item, factor], "\n"
    )
    loadings[i] <- L[item, factor]
    loading_names[i] <- paste0("lambda", i)
    i <- i + 1
  }
}

# Specify latent variable variances (fixed at 1)
latent_var_spec <- ""
latent_var_names <- colnames(L)
for (factor in seq_len(ncol(L))) {
  latent_var_spec <- paste0(
    latent_var_spec, "F", factor,
    " <-> ", "F", factor, ", NA, 1\n"
  )
}

# Specify latent variable correlations
latent_cor_spec <- ""
latent_cor <- NULL
latent_cor_names <- NULL
if (ncol(L) > 1) {
  var_pairs <- utils::combn(nrow(Phi), 2)
  latent_cor <- numeric(length = ncol(var_pairs))
  latent_cor_names <- character(length = ncol(var_pairs))
  for (pair in seq_len(ncol(var_pairs))) {
    fi <- var_pairs[1, pair]
    fj <- var_pairs[2, pair]
    latent_cor_spec <- paste0(
      latent_cor_spec, "F", fi, " <-> ", "F", fj,
      ", phi", fi, fj, ", ", Phi[fi, fj], "\n"
    )
    latent_cor[pair] <- Phi[fi, fj]
    latent_cor_names[pair] <- paste0("phi", fi, fj)
  }
}
```

```
  # Specify observed variable variances
  obs_var_spec <- ""
  obs_var <- numeric(length = nrow(L))
  obs_var_names <- paste0("psi", seq_len(nrow(L)))
  for (item in seq_len(nrow(L))) {
    obs_var_spec <- paste0(
      obs_var_spec, "V", item, " <-> ", "V", item, ", psi", item, ", ",
      u[item], "\n"
    )
    obs_var[item] <- u[item]
  }

  # Combine the loading, factor variance, and factor correlation specifications
  # to form the complete model specification
  model <- paste(
    loading_spec,
    latent_var_spec,
    latent_cor_spec,
    obs_var_spec, "\n"
  )

  # Vector of model parameters
  theta <- c(loadings, latent_cor, obs_var)
  names(theta) <- c(loading_names, latent_cor_names, obs_var_names)

  # Return sem model, vector of (named) model parameters, and factor names
  list(
    model = sem::specifyModel(text = model, quiet = TRUE),
    theta = theta,
    latent_var = latent_var_names
  )
}
#' Optimize TKL parameters to find a solution with target RMSEA and CFI values
#'
#' Find the optimal W matrix such that the RMSEA and CFI values are as close as
#' possible to the user-specified target values.
#'
#' @param mod A `fungible::simFA()` model object.
#' @param target_rmsea (scalar) Target RMSEA value.
#' @param target_cfi (scalar) Target CFI value.
#' @param tkl_ctrl (list) A control list containing the following TKL-specific
#'   arguments:
#'   * weights (vector) Vector of length two indicating how much weight to give
#'   RMSEA and CFI, e.g., `c(1,1)` (default) gives equal weight
#'   to both indices; `c(1,0)` ignores the CFI value.
#'   * v_start (scalar) Starting value to use for \eqn{\upsilon}, the proportion
#'   of uniqueness variance reallocated to the minor common factors. Note that
#'   only `v` as a proportion of the unique (not total) variance is supported
#'   in this function.
#'   * eps_start (scalar) Starting value to use for \eqn{\epsilon}, which
#'   controls how common variance is distributed among the minor common factors.
#'   * NminorFac (scalar) Number of minor common factors.
#'   * WmaxLoading (scalar) Threshold value for `NWmaxLoading`.
#'   * NWmaxLoading (scalar) Maximum number of absolute loadings \eqn{\ge}
#'   `WmaxLoading` in any column of \eqn{W}.
#'   * penalty (scalar) Penalty applied to objective function if the
#'   `NmaxLoading` condition isn't satisfied.
#'   * optim_type (character)  Which optimization function to use, `optim` or
#'   `ga`? `optim()` is faster, but might not converge in some cases. If `optim`
#'    doesn't converge, `ga` will be used as a fallback option.
#'   * max_tries (numeric) How many times to restart optimization with new start
#'   parameter values if optimization doesn't converge?
#'   * factr (numeric) controls the convergence of the "L-BFGS-B" method.
#'   Convergence occurs when the reduction in the objective is within this
#'   factor of the machine tolerance. Default is 1e7, that is a tolerance of
#'   about 1e-8. (when using `optim`).
#'   * maxit (number) Maximum number of iterations to use (when using `optim`).
#'   * ncores (boolean/scalar) Controls whether `ga()` optimization is done in
#'   parallel. If `TRUE`, uses the maximum available number of processor cores.
#'   If `FALSE`, does not use parallel processing. If an integer is provided,
#'   that's how many processor cores will be used (if available).
#' @md
#'
#' @details This function attempts to find optimal values of the TKL parameters
#'   \eqn{\upsilon} and \eqn{\epsilon} such that the resulting correlation
#'   matrix with model error (\eqn{\Sigma}) has population RMSEA and/or CFI
#'   values that are close to the user-specified values. It is important to note
#'   that solutions are not guaranteed to produce RMSEA and CFI values that are
#'   reasonably close to the target values; in fact, some combinations of RMSEA
#'   and CFI will be difficult or impossible to obtain for certain models (see
#'   Lai & Green, 2016). It can be particularly difficult to find good solutions
#'   when additional restrictions are placed on the minor factor loadings (i.e.,
#'   using the `WmaxLoading` and `NWmaxLoading` arguments).
#'
#'   Optimization is fastest when the `optim_type = optim` optimization method
#'   is chosen. This indicates that optimization should be done using the
#'   `L-BFGS-B` algorithm implemented in the `optim()` function. However, this
#'   method can sometimes fail to find a solution. In that case, I recommend
#'   setting `optim_type = ga`, which indicates that a genetic algorithm
#'   (implemented in `GA::ga()`) will be used. This method takes longer than
#'   `optim()` but is more likely to find a solution.
#'
#' @export
#' @references Tucker, L. R., Koopman, R. F., & Linn, R. L. (1969). Evaluation
#'   of factor analytic research procedures by means of simulated correlation
#'   matrices. *Psychometrika*, *34*(4), 421-459.
#'   <https://doi.org/10/chcxvf>

tkl <- function(mod,
                target_rmsea = NULL,
                target_cfi = NULL,
                tkl_ctrl = list()) {

  # Create default tkl_ctrl list; modify elements if changed by the user
  tkl_ctrl_default <- list(weights = c(rmsea = 1, cfi = 1),
                           v_start = stats::runif(1, 0.02, 0.9),
                           eps_start = stats::runif(1, 0, 0.8),
                           NMinorFac = 50,
                           WmaxLoading = NULL,
                           NWmaxLoading = 2,
                           debug = FALSE,
                           penalty = 1e6,
                           optim_type = "optim",
                           max_tries = 100,
                           factr = 1e6,
                           maxit = 5000,
                           ncores = FALSE)

  # Update the elements of the default tkl_ctrl list that have been changed by
  # the user
  tkl_ctrl_default <- tkl_ctrl_default[sort(names(tkl_ctrl_default))]
  tkl_ctrl_default[names(tkl_ctrl)] <- tkl_ctrl

  # Create objects for each of the elements in tkl_ctrl
  weights <- tkl_ctrl_default$weights
  v_start <- tkl_ctrl_default$v_start
  eps_start <- tkl_ctrl_default$eps_start
  NMinorFac <- tkl_ctrl_default$NMinorFac
  WmaxLoading <- tkl_ctrl_default$WmaxLoading
  NWmaxLoading <- tkl_ctrl_default$NWmaxLoading
  debug <- tkl_ctrl_default$debug
  penalty <- tkl_ctrl_default$penalty
  optim_type <- tkl_ctrl_default$optim_type
  ncores <- tkl_ctrl_default$ncores
  max_tries <- tkl_ctrl_default$max_tries
  factr <- tkl_ctrl_default$factr

  # Check arguments
  if (!is.null(target_rmsea)) {
    if (target_rmsea < 0 | target_rmsea > 1) {
      stop("The target RMSEA value must be a number between 0 and 1.\n",
           crayon::cyan("\u2139"), " You've specified a target RMSEA value of ",
           target_rmsea, ".", call. = F)
    }
  }
  if (!is.null(target_cfi)) {
    if (target_cfi > 1 | target_cfi < 0) {
      stop("Target CFI value must be between 0 and 1\n",
           crayon::cyan("\u2139"), " You've specified a target CFI value of ",
           target_cfi, ".", call. = F)
    }
  }
  if (is.null(target_cfi) & is.null(target_rmsea)) {
    stop("Either target RMSEA or target CFI (or both) must be specified.")
  }
  if (eps_start < 0 | eps_start > 1) {
    stop("The value of eps_start must be between 0 and 1.", call. = F)
  }
  if (v_start < 0 | v_start > 1) {
    stop("The value of v_start must be between 0 and 1.", call. = F)
  }
  if (!(is.list(mod)) |
      is.null(mod$loadings) |
      is.null(mod$Phi) |
      is.null(mod$Rpop)) {
    stop("`mod` must be a valid `simFA()` model object.", call. = F)
  }
  if (!is.numeric(weights) | length(weights) != 2) {
    stop("`weights` must be a numeric vector of length two.", call. = F)
  }
  if (NMinorFac < 0) {
    stop("The number of minor factors must be non-negative.\n",
         crayon::cyan("\u2139"), " You've asked for ", NMinorFac,
         " minor factors.", call. = F)
  }
  if (!(optim_type %in% c("optim", "ga"))) {
    stop("`optim_type` must be either `optim` or `ga`.\n",
         crayon::cyan("\u2139"), " You've supplied ", optim_type,
         " as `optim_type`.", call. = F)
  }
  if (!is.numeric(penalty) | penalty < 0) {
    stop("`penalty` must be a postive number.\n",
         crayon::cyan("\u2139"), " You've supplied ", penalty,
         " as `penalty`.", call. = F)
  }
  if (!is.null(WmaxLoading)) {
    if (!is.numeric(WmaxLoading) | WmaxLoading <= 0) {
      stop("`WmaxLoading` must be a positive number.\n",
           crayon::cyan("\u2139"), " You've supplied ", WmaxLoading,
           " as `WmaxLoading`.", call. = F)
    }
  }
```

```r
if (((NWmaxLoading %% 1) != 0) | NWmaxLoading < 0) {
  stop("`NWmaxLoading` must be a non-negative integer.\n",
       crayon::cyan("\u2139"), " You've supplied ", NWmaxLoading,
       " as `NWmaxLoading`.", call. = F)
}

# If no CFI value is given, set the weight to zero and set target_cfi to a
# no-null value (it will be ignored in the optimization)
if (is.null(target_cfi)) {
  weights[2] <- 0
  target_cfi <- 999
}

# Same for RMSEA
if (is.null(target_rmsea)) {
  weights[1] <- 0
  target_rmsea <- 999
}

L <- mod$loadings
Phi <- mod$Phi

# Create W with eps = 0
W <- MASS::mvrnorm(
  n = nrow(L),
  mu = rep(0, NMinorFac),
  Sigma = diag(NMinorFac)
)

p <- nrow(L) # number of items
k <- ncol(L) # number of major factors

CovMajor <- L %*% Phi %*% t(L)
u <- 1 - diag(CovMajor)
Rpop <- CovMajor
diag(Rpop) <- 1 # ensure unit diagonal

df <- (p * (p - 1) / 2) - (p * k) + (k * (k - 1) / 2) # model df

start_vals <- c(v_start, eps_start)

if (optim_type == "optim") {
  ctrl <- list(factr = factr)
  if (debug == TRUE) {
    ctrl$trace <- 5
    ctrl$REPORT <- 1
  }
  # Try optim(); if it fails, then use GA instead
```

```
    opt <- NULL
    tries <- 0
    converged <- FALSE
    while (converged == FALSE & (tries <= max_tries)) {
      if (tries > 1) start_vals <- c(v_start = stats::runif(1, 0.02, 0.9),
                                     eps_start = stats::runif(1, 0, 0.8))
      tryCatch(
        {
          opt <- stats::optim(
            par = start_vals,
            fn = obj_func,
            method = "L-BFGS-B",
            lower = c(0.001, 0), # can't go lower than zero
            upper = c(1, 1), # can't go higher than one
            Rpop = Rpop,
            W = W,
            p = p,
            u = u,
            df = df,
            target_rmsea = target_rmsea,
            target_cfi = target_cfi,
            weights = weights,
            WmaxLoading = WmaxLoading,
            NWmaxLoading = NWmaxLoading,
            control = ctrl,
            penalty = penalty
          )
          par <- opt$par
        },
        error = function(e) NULL
      )

      tries <- tries + 1
      if (is.null(opt)) {
        converged <- FALSE
        next
      }

      converged <- opt$convergence == 0
    }

    # If the algorithm fails to converge or produces NULL output, try GA instead
    if (is.null(opt)) {
      opt <- list(convergence = FALSE)
    }

    if (opt$convergence != 0) {
      optim_type <- "ga"
      warning("`optim()` failed to converge, using `ga()` instead.",
              call. = FALSE)
    }
  }

  if (optim_type == "ga") {
    opt <- GA::ga(
      type = "real-valued",
      fitness = function(x) {
        -obj_func(x,
          Rpop = Rpop,
          W = W,
          p = p,
          u = u,
          df = df,
          target_rmsea = target_rmsea,
          target_cfi = target_cfi,
          weights = weights,
          WmaxLoading = WmaxLoading,
          NWmaxLoading = NWmaxLoading,
          penalty = penalty
        )
      },
      lower = c(0, 0),
      upper = c(1, 1),
      popSize = 50,
      maxiter = 1000,
      run = 100,
      parallel = ncores,
      monitor = FALSE
    )
    par <- opt@solution[1, ]
  }

  obj_func(
    par = par,
    Rpop = Rpop,
    W = W,
    p = p,
    u = u,
    df = df,
    target_rmsea = target_rmsea,
    target_cfi = target_cfi,
    weights = weights,
    WmaxLoading = WmaxLoading,
    NWmaxLoading = NWmaxLoading,
    return_values = TRUE,
    penalty = penalty
  )
}
#' Wu & Browne model error method
#'
#' Generate a population correlation matrix using the model described in Wu and
#' Browne (2015).
#'
#' @param mod A `fungible::simFA()` model object.
#' @param target_rmsea (scalar) Target RMSEA value.
#' @param wb_mod (`lm` object) An optional `lm` object used to find a target
#'   RMSEA value that results in solutions with RMSEA values close to the
#'   desired value. Note that if no `wb_mod` is provided, a model will be
#'   estimated at run time. If many population correlation matrices are going to
#'   be simulated using the same model, it will be considerably faster to
#'   estimate `wb_mod` ahead of time. See also `get_wb_mod()`.
#' @param adjust_target (TRUE; logical) Should the target_rmsea value be
#'   adjusted to ensure that solutions have RMSEA values that are close to the
#'   provided target RMSEA value? Defaults to TRUE and should stay there unless
#'   you have a compelling reason to change it.
#'
#' @author Justin Kracht <krach018@umn.edu>
#' @references Wu, H., & Browne, M. W. (2015). Quantifying adventitious error in
#'   a covariance structure as a random effect. *Psychometrika*, *80*(3),
#'   571-600. <https://doi.org/10/gjrkc4>
#'
#' @export
#' @details The Wu and Browne method generates a correlation matrix with model
#'   error (\eqn{\Sigma}) using
#'
#'   \deqn{(\Sigma | \Omega) ~ IW(m, m \Omega),}
#'
#'   where \eqn{m ~= 1/\epsilon^2} is a precision parameter related to RMSEA
#'   (\eqn{\epsilon}) and \eqn{IW(m, m \Omega)} denotes an inverse Wishart
#'   distribution. Note that *there is no guarantee that the RMSEA will be very
#'   close to the target RMSEA*, particularly when the target RMSEA value is
#'   large. Based on experience, the method tends to give solutions with RMSEA
#'   values that are larger than the target RMSEA values. Therefore, it might be
#'   worth using a target RMSEA value that is somewhat lower than what is
#'   actually needed. Alternatively, the \code{\link{get_wb_mod}} function can
#'   be used to estimate a coefficient to shrink the target RMSEA value by an
#'   appropriate amount so that the solution RMSEA values are close to the
#'   (nominal) target values.
#'
#' @examples
#' # Specify a default model using simFA()
#' mod <- fungible::simFA(Seed = 42)
#'
#' set.seed(42)
```

```r
#' wb(mod, target_rmsea = 0.05)

wb <- function(mod,
               target_rmsea,
               wb_mod = NULL,
               adjust_target = TRUE) {

  if (!(is.list(mod)) |
      is.null(mod$loadings) |
      is.null(mod$Phi) |
      is.null(mod$Rpop)) {
    stop("`mod` must be a valid `simFA()` model object.", call. = F)
  }
  if (target_rmsea < 0 | target_rmsea > 1) {
    stop("The target RMSEA value must be a number between 0 and 1.\n",
         crayon::cyan("\u2139"), " You've specified a target RMSEA value of ",
         target_rmsea, ".", call. = F)
  }
  if (!is.null(wb_mod)) {
    if (class(wb_mod) != "lm") {
      stop("`wb_mod` must be an object of class `lm`.", call. = F)
    }
  }

  if (is.null(wb_mod) & (adjust_target == TRUE)) {
    if (target_rmsea >= 0.095) {
      upper <- target_rmsea + 0.01
    } else {
      upper <- 0.095
    }
    wb_mod <- get_wb_mod(mod, upper = upper)
  }

  # Use wb_mod to find the correct target_rmsea value to use
  if (!is.null(wb_mod) & (adjust_target == TRUE)) {
    target_rmsea <- stats::predict(
      wb_mod,
      newdata = data.frame(rmsea_medians = target_rmsea)
    )
  }

  v <- target_rmsea^2
  m <- v^-1 # m is the precision parameter, Wu and Browne (2015), p. 576

  Omega <- mod$Rpop
  p <- nrow(Omega)
  if (m < p) {
    stop("Target RMSEA value is too large, try a smaller value.", call. = FALSE)
  }

  Sigma <- MCMCpack::riwish(m, m * Omega)
  list(Sigma = stats::cov2cor(Sigma),
       m = m)
}
```

A.2 Simulation Code

A.2.1 Data Generation

```r
if (!require(noisemaker)) {
  devtools::install_github("JustinKracht/noisemaker",
                           dependencies = TRUE)
}

library(noisemaker)
library(here)
library(fungible)
library(tidyverse)
library(pbmcapply)

# Set variable values -----------------------------------------------------

DATA_DIR <- here("data")

# Number of simulation reps
reps <- 500

# Create a matrix of all of the simulation conditions
condition_matrix <- tidyr::expand_grid(
  factors = c(1, 3, 5, 10),
  items_per_factor = c(5, 15),
  factor_cor = c(0, .3, .6),
  loading = c("weak", "moderate", "strong"),
  target_rmsea = c(0.025, 0.065, 0.090)
) %>% filter(
  !(factors == 1 & factor_cor != 0)
) %>% mutate(
  condition_num = 1:n()
) %>% dplyr::relocate(condition_num)

# Add target CFI values
condition_matrix <- condition_matrix %>% mutate(
  target_cfi = case_when(target_rmsea == 0.025 ~ 0.99,
                         target_rmsea == 0.065 ~ 0.95,
                         target_rmsea == 0.090 ~ 0.90)
)

saveRDS(condition_matrix, here("data", "condition_matrix.RDS"))

# Make matrix of optimization method choices for TKL
method_matrix <- tidyr::expand_grid(
  rmsea_weight = c(0,1),
  cfi_weight = c(0,1)
) %>% filter(rmsea_weight != 0 | cfi_weight != 0)

# tryCatch function to wrap noisemaker so that warnings and errors are captured
myTryCatch <- function(expr) {
  warn <- err <- NULL
  value <- withCallingHandlers(
    tryCatch(expr, error=function(e) {
      err <<- e
      NULL
    }), warning=function(w) {
      warn <<- w
      invokeRestart("muffleWarning")
    })
  list(value=value, warning=warn, error=err)
}

# Simulation loop ---------------------------------------------------------

RNGkind("L'Ecuyer-CMRG")
set.seed(666)
seed_list <- sample(1e7, size = nrow(condition_matrix), replace = FALSE)

results_list <- pbmclapply(
  X = seq_along(condition_matrix$condition_num),
  # X = 154:nrow(condition_matrix),
  FUN = function(condition) {

    set.seed(seed_list[condition])

    factors <- condition_matrix$factors[condition]
    items_per_factor <- condition_matrix$items_per_factor[condition]
    factor_cor <- condition_matrix$factor_cor[condition]
    loading <- condition_matrix$loading[condition]
    target_rmsea <- condition_matrix$target_rmsea[condition]
    target_cfi <- condition_matrix$target_cfi[condition]

    FacLoadRange <- switch(loading,
                           "weak" = 0.4,
                           "moderate" = 0.6,
                           "strong" = 0.8)

    # Generate factor model
    mod <- simFA(Model = list(NFac = factors,
                              NItemPerFac = items_per_factor,
                              Model = "oblique"),
                 Loadings = list(FacLoadDist = "fixed",
                                 FacLoadRange = FacLoadRange),
                 Phi = list(PhiType = "fixed",
                            MaxAbsPhi = factor_cor))

    wb_mod <- get_wb_mod(mod, n = 100, values = 15)

    sigma_list <- purrr::map(
      .x = seq_len(reps),
      .f = function(i, mod, target_rmsea) {

        # TKL method variations
        sigma_tkl_rmsea <- myTryCatch(
          noisemaker(
            mod,
            method = "TKL",
            target_rmsea = target_rmsea,
            target_cfi = NULL,
            tkl_ctrl = list(WmaxLoading = .3,
                            NWmaxLoading = 2,
                            max_tries = 100,
                            maxit = 1000)
          )
        )

        sigma_tkl_cfi <- myTryCatch(
          noisemaker(
            mod,
            method = "TKL",
            target_rmsea = NULL,
            target_cfi = target_cfi,
            tkl_ctrl = list(WmaxLoading = .3,
                            NWmaxLoading = 2,
                            max_tries = 100,
                            maxit = 1000)
          )
        )

        sigma_tkl_rmsea_cfi <- myTryCatch(
          noisemaker(
            mod,
            method = "TKL",
            target_rmsea = target_rmsea,
            target_cfi = target_cfi,
            tkl_ctrl = list(WmaxLoading = .3,
                            NWmaxLoading = 2,
                            max_tries = 100,
                            maxit = 1000)
          )
        )

        # Other methods
        # Skip CB for the largest conditions; takes way too long.
        if (condition %in% 154:nrow(condition_matrix)) {
          sigma_cb <- list("value" = NA,
                           "warning" = NULL,
                           "error" = NULL)
        } else {
          sigma_cb  <- myTryCatch(noisemaker(mod, method = "CB",
                                             target_rmsea = target_rmsea))
        }
        sigma_wb  <- myTryCatch(noisemaker(mod, method = "WB",
                                           target_rmsea = target_rmsea,
                                           wb_mod = wb_mod))

        list(sigma_tkl_rmsea = sigma_tkl_rmsea,
             sigma_tkl_cfi = sigma_tkl_cfi,
             sigma_tkl_rmsea_cfi = sigma_tkl_rmsea_cfi,
             sigma_cb   = sigma_cb,
             sigma_wb   = sigma_wb)
      },
      mod = mod,
      target_rmsea = target_rmsea
    )

    # Save condition results
    saveRDS(
      sigma_list,
      file = here(
        "data",
        paste0("results_",
               formatC(condition, width = 3, flag = 0),
               ".RDS"
        )
      )
    )

  },
  mc.preschedule = FALSE,
  mc.cores = 18
)
# Extract simulation data from lists and calculate alternative fit statistics

library(dplyr)
library(tidyr)
library(purrr)
library(parallel)
library(stringr)
library(fungible)
library(here)
library(purrrgress)

condition_matrix <- readRDS(here("data", "condition_matrix.RDS"))

# Create results matrix ---------------------------------------------------

reps <- 500

# Create a matrix to hold results
results <- condition_matrix[rep(1:nrow(condition_matrix), each = reps),]

# For each rep in each condition, record the rep number
results$rep_num <- rep(1:reps, times = nrow(condition_matrix))

# Create model error method variable
results <- expand_grid(
  results,
  error_method = c("TKL_rmsea", "TKL_cfi", "TKL_rmsea_cfi", "CB", "WB")
)

# Function definitions ----------------------------------------------------

# Compute the fit statistics that were not computed in the main simulation
# loop

compute_fit_stats <- function(error_method_results, Rpop, k) {

  # Check if value is NULL; if so, return empty list
  if (is.null(error_method_results$value)) {
    out = data.frame(RMSEA_thetahat = NA,
                     CFI_thetahat = NA,
                     CRMR_theta = NA,
                     CRMR_thetahat = NA,
                     TLI_theta = NA,
                     TLI_thetahat = NA)
    return(out)
  }
```

```r
RpopME <- pluck(error_method_results, "value", "Sigma")

p <- ncol(RpopME)
tp <- p * (p + 1)/2

fout <- factanal(covmat = RpopME, factors = k,
                 rotation = "none", n.obs = 1000,
                 control = list(nstart = 100))

Fhat <- fout$loadings
Sigma_k <- Fhat %*% t(Fhat)
diag(Sigma_k) <- 1

Tr <- function(X) sum(diag(X))

num_thetahat <- log(det(Sigma_k)) - log(det(RpopME)) +
  Tr(RpopME %*% solve(Sigma_k)) - p

k <- ncol(Fhat)
DF <- (p * (p - 1)/2) - (p * k) + (k * (k - 1)/2)
DF_B <- (p * (p - 1)/2) - (p * p) + (p * (p - 1)/2)

F_T_thetahat <- log(det(Sigma_k)) - log(det(RpopME)) +
  Tr(RpopME %*% solve(Sigma_k)) - p
F_B_thetahat <- -log(det(RpopME))

F_T_theta <- log(det(Rpop)) - log(det(RpopME)) +
  Tr(RpopME %*% solve(Rpop)) - p
F_B_theta <- -log(det(RpopME))

# Calculate RMSEA thetahat
RMSEA_thetahat <- sqrt(num_thetahat/DF)

# Calculate CFI thetahat
CFI_thetahat <- 1 - F_T_thetahat/F_B_thetahat

# Calculate CRMR values
CRMR_theta <- sqrt(
  sum(((RpopME - Rpop)[upper.tri(Rpop, diag = FALSE)]^2)) / (tp - p)
)

CRMR_thetahat <- sqrt(
  sum(((RpopME - Sigma_k)[upper.tri(Rpop, diag = FALSE)]^2)) / (tp - p)
)

# Calculate TLI values (from Xia & Yang, 2019, p. 412)
TLI_theta <- 1 - ( (F_T_theta / DF) / (F_B_theta / DF_B) )
TLI_thetahat <- 1 - ( (F_T_thetahat / DF) / (F_B_thetahat / DF_B) )
```

```r
  data.frame(RMSEA_thetahat = RMSEA_thetahat,
             CFI_thetahat = CFI_thetahat,
             CRMR_theta = CRMR_theta,
             CRMR_thetahat = CRMR_thetahat,
             TLI_theta = TLI_theta,
             TLI_thetahat = TLI_thetahat)
}

# Compute d values
compute_d_values <- function(error_method_data, target_rmsea, target_cfi) {

  if (is.null(error_method_data$value)) {
    out = data.frame(d1 = NA,
                     d2 = NA,
                     d3 = NA)
    return(out)
  }

  rmsea <- pluck(error_method_data, "value", "rmsea")
  cfi <- pluck(error_method_data, "value", "cfi")

  d1 <- abs(rmsea - target_rmsea)
  d2 <- abs(cfi - target_cfi)
  d3 <- d1 + d2

  data.frame(
    d1 = d1,
    d2 = d2,
    d3 = d3
  )
}

# Check if there are any major factors in W
check_w_major_factors <- function(W) {
  if (!is.matrix(W)) {
    NA
  } else {
    sum(W[,1] >= .3) > 2
  }
}

# Create a vector of result file paths ------------------------------------

results_files <- list.files(
  path = "data",
  pattern = ".*[0-9]+\\.RDS",
  full.names = TRUE
)
```

```
# Read data from each condition and calculate statistics ------------------

# Calculate alternative fit statistics

pbmcapply::pbmclapply(
  X = seq_along(results_files),
  FUN = function(i, results_files, condition_matrix) {

    cat("/nWorking on condition:", i)

    # Read in loading matrix
    condition_results <- readRDS(results_files[i])

    # Extract the condition number from the results file name
    condition_num <- as.numeric(
      str_extract(results_files[i], pattern = "[0-9]+")
    )

    # Extract condition information (loading strength, num. factors, num. items)
    j <- which(condition_matrix$condition_num == condition_num)
    loading_condition <- condition_matrix$loading[j]
    k <- condition_matrix$factors[j]
    p <- condition_matrix$items_per_factor[j] * k

    # Extract target RMSEA and CFI values
    target_rmsea <- condition_matrix$target_rmsea[j]
    target_cfi <- condition_matrix$target_cfi[j]

    # Create factor loading matrix
    loadings <- switch(loading_condition,
                       "weak" = .4,
                       "moderate" = .6,
                       "strong" = .8)

    # Extract Rpop
    mod <- fungible::simFA(Model = list(NFac = condition_matrix$factors[j],
                                        NItemPerFac = condition_matrix$items_per_factor[j],
                                        Model = "oblique"),
                           Loadings = list(FacLoadDist = "fixed",
                                           FacLoadRange = loadings),
                           Phi = list(PhiType = "fixed",
                                      MaxAbsPhi = condition_matrix$factor_cor[j]))
    Rpop <- mod$Rpop

    condition_results_matrix <- purrr::map_dfr(
      .x = seq_along(condition_results),
      .f = function(z, Rpop, p, k, target_rmsea, target_cfi,
                    condition_num) {
```

```
        rep_results <- condition_results[[z]]
        other_fit_stats <- map_dfr(.x = rep_results,
                                   ~ compute_fit_stats(., Rpop = Rpop, k = k))
        d_values <- map_dfr(.x = rep_results,
                            ~ compute_d_values(.,
                                               target_rmsea = target_rmsea,
                                               target_cfi = target_cfi))

        out <- data.frame(
          condition_num = rep(condition_num, 5),
          rep_num = rep(z, 5),
          cfi = map_dbl(rep_results, ~ pluck(., "value", "cfi", .default = NA)),
          cfi_thetahat = other_fit_stats$CFI_thetahat,
          rmsea = map_dbl(rep_results, ~ pluck(., "value", "rmsea", .default = NA)),
          rmsea_thetahat = other_fit_stats$RMSEA_thetahat,
          crmr = other_fit_stats$CRMR_theta,
          crmr_thetahat = other_fit_stats$CRMR_thetahat,
          tli = other_fit_stats$TLI_theta,
          tli_thetahat = other_fit_stats$TLI_thetahat,
          m = map_dbl(rep_results, ~ pluck(., "value", "m", .default = NA)),
          v = map_dbl(rep_results, ~ pluck(., "value", "v", .default = NA)),
          eps = map_dbl(rep_results, ~ pluck(., "value", "eps", .default = NA)),
          fn_value = map_dbl(rep_results, ~ pluck(., "value", "fn_value", .default = NA)),
          error_method = c("tkl_rmsea", "tkl_cfi", "tkl_rmsea_cfi", "cb", "wb"),
          w_has_major_factors = map_lgl(rep_results,
                                        ~ check_w_major_factors(
                                          pluck(., "value", "W", .default = NA)
                                        )),
          d1 = d_values$d1,
          d2 = d_values$d2,
          d3 = d_values$d3,
          warning = map_chr(rep_results, .f = function(x) {
            warning <- pluck(x, "warning")
            if (is.null(warning)) {
              warning <- NA
            } else {
              warning <- as.character(warning)
            }
            warning
          }),
          error = map_chr(rep_results, .f = function(x) {
            error <- pluck(x, "error")
            if (is.null(error)) {
              error <- NA
            } else {
              error <- as.character(error)
            }
            error
          })
        )

        rownames(out) <- NULL
        out
      },
      Rpop = Rpop, p = p, k = k, target_rmsea = target_rmsea,
      target_cfi = target_cfi, condition_num
    )

    saveRDS(condition_results_matrix,
            file = paste0("data/results_matrix_",
                          formatC(i, width = 3, flag = 0),
                          ".RDS"))

  }, results_files = results_files, condition_matrix = condition_matrix,
  mc.cores = 8
)
```

Appendix B

Supplemental Tables and Figures

Here is where we can put extra material that is useful, but perhaps not critical to the overall paper. This could include instruments, consent forms, additional syntax, proofs, supplemental graphics, etc.