Assessing local fit by approximating probabilities

REMOVED FOR PEER REVIEW

REMOVED FOR PEER REVIEW

Author Note

REMOVED FOR PEER REVIEW

Abstract

Validity evidence for factor structures underlying a set of items can come from how well a proposed model reconstructs, or fits, the observed relationships. Global model fit is limited in that some components of the proposed model fit better than other components. This limitation has lead to the recommendation of examining fit locally within model components. We describe a new probabilistic approach to assessing local fit using a Bayesian approximation, and illustrate use with a simulated dataset. We show how the posterior approximation closely approximated the sampling distribution of the true parameter. We discuss potential limitation and possible generalizations.

Keywords: local fit, factor analysis, Laplace, posterior probability

Assessing local fit by approximating probabilities

Statistical models, such as structural equation models, are intended to serve as an approximation of the process or mechanisms that generated the observed data. The approximation of reality has been described as arising from distributional and structural approximations (Bollen, 2019). When models fail to closely approximate reality (i.e., misfit), identifying the source(s) of model misspecifications can be at best, difficult, and at worst, impossible. Fortunately, a variety of methods are available to aid researchers in evaluating the fit of their proposed model for measuring one or more constructs (DiStefano, 2016). In fact, statistical information that demonstrates acceptable model-data fit can be taken as evidence for a measurement models construct validity.

One popular method of collecting validity evidence supporting one's proposed measurement model involves the use of global fit indices. Global fit indices help identify whether the specified measurement model adequately approximates the data generating process overall. Commonly used global fit indices are the comparative fit index (Bentler, 1990), root mean square error of approximation (Browne & Cudeck, 1992), and standardized root mean square residual (Bentler, 1995; Jöreskog & Sörbom, 1981; Maydeu-Olivares et al., 2018). Many of these indices are now automatically computed in most structural equation modeling software. However, these indices only provide a coarse estimate of overall model-data fit (Steiger, 2007).

One difficulty arising from the coarse nature of global fit indices is that when the indices are below acceptable values a researcher must explain the lack of fit. Global fit indices do not necessarily help identify where the misspecification is occurring though. Some parts of the model may better represent the latent structure than other parts, but identifying which parts are not representative is not straightforward. Methods of identifying the source of misfit include investigating residual matrices (Kline, 2015; Maydeu-Olivares & Shi, 2017), modification indices (Kaplan, 1989; Sörbom, 1989), Wald tests (Buse, 1982; Wald, 1943), likelihood ratio tests (Buse, 1982; Neyman & Pearson,

1928), and model-implied instrumental variables (Bollen, 1995, 2019). Each method has benefits and pitfalls that need to be considered when looking for evidence of model-data fit. The methods above generally provide users with a way of identifying which specific relationships among observed variables are not captured by the model (e.g., residuals) or which specific component of the model does not fit well (e.g., Wald tests). Once a change in the model has been made, a researcher can then compare the results of local fit to see if fit improved.

Evaluating the change in model fit has been helpful for many researchers in the model fit evaluation process as evident by the prolonged use of these approaches, but this process has potential drawbacks. For example, modification indices can produce conflicting information about sources of misfit, which can in turn lead to statistically equivalent models. Adding too many paths to the measurement or structural components of the model may result in difficulty interpreting model parameters due to model overfit. Aside from investigating standardized residuals (Maydeu-Olivares & Shi, 2017), most current methods of investigating local fit rely on potentially non-intuitive metrics, such as modification indices, likelihood ratios, or p-values. In order to improve the interpretability of local fit investigations, we propose an approach utilizing probabilities as the inferential unit for local fit assessment. In the proposed probabilistic approach, the aim is to approximate whether additional paths will result in a meaningfully interpretable contribution to the model. Our method helps control against overfit through interpretation-driven local fit assessment.

There are numerous important contributions of this work. First, we extend previous research on local fit assessment by introducing a probabilistic method utilizing Bayesian methods. Second, we demonstrate the application of this approach in a simulated dataset with a known latent structure. Third, we investigated the performance through Monte Carlo simulation study evaluating the accuracy of the predicted probabilities. Lastly, we discuss the benefits and potential pitfalls of the new proposed local fit assessment.

Proposed probabilistic method for local fit assessment

Local fit assessment by means of investigating residual matrices, modification indices, Wald tests, likelihood ratio tests or model-implied instrumental variables have each shown to provide useful information in a variety of contexts (Chou & Bentler, 1990; Maydeu-Olivares & Shi, 2017; Whittaker, 2012). However, these methods fail to reflect whether potential model modifications would result in a change in substantive interpretation of the model. To address this issue, we have developed a relatively straightforward approach to evaluate local fit by means of an iterative approximation of the non-estimated parameters.

The proposed method approximates the magnitude of the non-estimated parameters if added to the model, where magnitude is approximated by a probability distribution to represent the uncertainty in the parameter. Using the distribution of each parameter, a probability can be computed to capture how likely that the parameter would be of substantive interest. The idea underlying this approach to local fit assessment is to define a region of the parameter space that would be considered meaningful for the substantive question at hand. A natural choice in the case of factor loadings is to define the region as $|\lambda| \geq 0.30$ (Benson & Nasser, 1998), which is the threshold typically considered as the bound for standardized factor loadings in exploratory factor analysis (EFA). However, we are not restricted to this region; for example, we might be interested in knowing if any unstandardized loadings would likely be greater than one, indicating that the item loads more strongly than the item we fixed for identification. This approach is similar to defining a "region of practical equivalence (ROPE)" described in Shi et al. (2019) as a Bayesian approach for measurement invariance testing. The ROPE is a region in the parameter space that the researcher determines to be insignificant, which is already common in most applications of EFA when the researcher suppresses factor loadings that are below 0.30 (Benson & Nasser, 1998).

The proposed to local fit assessment defines regions of the parameter space that are

of practical insignificance and then approximates the probability that our model parameters fall in that space. The probabilities are based on a rough approximation of a posterior distribution based on Bayes theorem. A similar quantity in Bayesian methods is the posterior predictive probability (PPP or p-value, Gelman et al., 1996; Rubin, 1996). The PPP is often used to construct probabilistic inferences about test statistics from the posterior predictive distribution. However, for the current application we are interested in computing a probability based on the posterior itself leading to a simpler approximation problem.

Lee et al. (2016) discussed a related approach whereby posterior approximation helps evaluate global model fit and identify influential observations. The authors suggested that the probabilistic approach can be used to assess model fit in a variety of ways such as residual covariances. Yet no one yet has built on this method for investigations of local fit assessment, and therefore, it is the focus of our work. Next, we briefly review the necessary aspects of Bayesian theory needed for the probability of inferential interest.

Bayesian methods underlying the assessment of local fit

Bayes theorem is one of the most powerful tools to manipulate probabilities. Bayes theorem provides a framework for decomposing a conditional probability into a likelihood statement about observed data and a hypothesized probability space for model parameters (i.e., the prior). This can be expressed:

$$p(\theta|Y) \propto \ell(Y|\theta)p(\theta),$$

where $\propto$ is read as "proportional to", $\ell(Y|\theta)$ is the likelihood of the data Y determined by the model specified and parameters θ, and $p(\theta)$ is the joint prior distribution for the parameters in the model. The posterior is usually sampled from using Monte Carlo methods (e.g., Gibbs sampling, Hamiltonian Monte Carlo, etc.). However, in this case we elect to utilize a *relatively* simpler approximation of the posterior from which we can sample from relatively quickly.

Under mild regularity conditions of the smoothness of the likelihood function, the posterior distribution for θ will approximate a normal distribution as the number of samples drawn from the distribution increases (Gelman et al., 2013). Using this property of the posterior, Laplace's method can be used to derive a second-order Taylor series approximation to the posterior (Tierney & Kadane, 1986). The Taylor series approximation provides a point estimate for the posterior expectation and variance. Then, using these two pieces matched with the knowledge that the posterior is asymptotically normally distributed, we arrive at an approximate posterior distribution for any parameter. The Laplace approximation has been shown to provide a useful approximation of posteriors for relatively simple models in a variety of contexts (Kass & Steffey, 1989; Rue et al., 2009).

Under the normal distribution approximation of the posterior, the Laplace approximation estimator, $\hat{\theta}$, is nearly equal to maximum likelihood estimator, $\hat{\theta}_{MLE}$. Similarly, the variance of the Laplace approximation estimator is nearly identical to the observed information matrix (I_{OBS}). Additionally, the use of the Laplace approximation has commonly excluded the prior distribution for θ in the approximation (Lee et al., 2016; Li, 1992; Rue et al., 2009; Wolfinger, 1993); however, excluding the prior decreases the uncertainty in the posterior distribution for θ even if only slightly. Additionally, utilizing a minimally informative prior structure provides a useful initializing point for numerical methods. Next, we will describe how Laplace's method can be applied in factor analysis for local fit assessment.

Applying the Laplace approximation to Bayesian CFA

Using the Bayesian factor analysis notation fromLevy and Choi (2013), the full model is

$$\begin{aligned} \mathbf{Y}_i &= \tau + \Lambda\eta_i + \theta_i \\ \Sigma &= \Lambda^T \Phi \Lambda + \Theta \\ \mathbf{Y}_i \mid \tau, \Lambda, \eta_i, \Theta &\sim \mathrm{N}\left(\tau + \Lambda\eta_i, \Theta\right) \\ \eta_i \mid \kappa, \Phi &\sim \mathrm{N}\left(\kappa, \Phi\right) \end{aligned} \tag{1}$$

where $\mathbf{Y}_i$ is the response vector for person i, τ is the vector of indicator intercepts, Λ is the matrix of factor loadings, η_i is the vector of factor scores, θ_i is the vector of residuals, Σ is the indicator/observed variable covariance matrix, Θ is the error covariance matrix, κ is the vector of factor intercepts (usually fixed to 0), and Φ is the factor covariance matrix. The full model could be incorporated in a Bayesian perspective relatively easily, but the parameter space grows rapidly. Below is one potential way the full joint posterior could be express.

$$\pi\left(\tau, \Lambda, \eta, \Theta, \kappa, \Phi \mid \mathrm{Y}\right) \propto \ell\left(\mathrm{Y} \mid \tau, \Lambda, \eta, \Theta, \kappa, \Phi\right) \times \pi\left(\tau, \Lambda, \eta, \Theta, \kappa, \Phi\right), \tag{2}$$

which increases in complexity as the large number of subjects, items, and latent variables increases. The likelihood is not difficult to define due to the assumption of conditional independence of subjects, and further simplification can be made if conditional independence of items is assumed (i.e., Θ is a diagonal matrix)

$$\begin{aligned} \ell\left(\mathrm{Y} \mid \tau, \Lambda, \eta, \Theta, \kappa, \Phi\right) &= \prod_{i=1}^{N} f\left(\mathbf{Y}_i \mid \tau, \Lambda, \eta_i, \Theta\right) \\ &= \prod_{i=1}^{N} \prod_{j=1}^{J} f\left(Y_{ij} \mid \tau_j, \lambda_j, \eta_i, \theta_{jj}\right). \end{aligned}$$

A more complicated issue is on the construction of the joint prior distribution because constructing joint prior distributions directly in multiparameter problems is typically not possible outside of special cases (e.g, normal-gamma conjugate prior in normal theory multiple linear regression). In multiparameter problems, a common method

for building the joint prior distribution is to construct the prior hierarchically. The hierarchical prior for factor models can be constructed by decomposing the prior into parameter groups that can defensibly be independent or at least conditionally independent. A potential decomposition is to have independent groups for factor loadings (Λ), item intercepts (τ), and error (co)variances (Θ), then group the factor scores (η), factor intercepts (κ), and factor covariance matrix (Φ) together. The later grouping can be structured where the factor scores distribution is conditional on the factor intercept and factor covariance matrix priors. Therefore, the decomposed prior could be expressed as

$$\begin{aligned}\pi\left(\tau, \Lambda, \eta, \Theta, \kappa, \Phi\right) &= \pi\left(\eta, \kappa, \Phi\right) \pi\left(\tau\right) \pi\left(\Lambda\right) \pi\left(\Theta\right) \\ &= \pi\left(\eta \mid \kappa, \Phi\right) \pi\left(\kappa\right) \pi\left(\Phi\right) \pi\left(\tau\right) \pi\left(\Lambda\right) \pi\left(\Theta\right).\end{aligned}$$

Some common specifications of the priors for these parameters are

$$\begin{aligned}\eta \mid \kappa, \Phi &\sim \mathrm{N}(\kappa, \Phi) \\ \kappa &\sim \mathrm{N}(\mu_\kappa, \sigma^2_\kappa) \\ \Phi &\sim \mathrm{Inverse-Wishart}(d\Phi_0, d),\ d \geq M \\ \tau_j &\sim \mathrm{N}(\mu_\tau, \sigma^2_\tau) \\ \lambda_{jm} &\sim \mathrm{N}(\mu_\lambda, \sigma^2_\lambda) \\ \theta_{jj} &\sim \mathrm{Inverse-Gamma}(\alpha_\theta, \beta_\theta) \\ \theta_{jk} &\sim \mathrm{N}(\mu_\theta, \sigma^2_\theta).\end{aligned}$$

The indices j correspond to the j^{th} indicator and m is index over latent variables. The factor loadings, intercepts, error (co)variances may be specified independently for each individual parameter which greatly simplifies the process, but is not necessary. If the error (co)variance matrix was not hypothesized to be diagonal, then that structure could be included as part of a joint prior on Θ.

Once the joint prior is constructed, it may be difficult in a full Bayesian analysis to sample from the posterior. A common solution is to employ some type of Markov Chain

Monte Carlo (MCMC) algorithm to efficiently sample from the marginal posteriors of each parameter. Several software options that make this job easier including blavaan (Merkle & Rosseel, 2018), JAGS (Plummer et al., 2003), M*plus* (Muthén & Muthén, 2017), OpenBUGS (Surhone et al., 2010), and Stan (Carpenter et al., 2017). That said, we propose a drastic simplification of the full Bayesian analysis with the aim of obtaining a rough approximation of a specific feature of the posterior.

As we have described previously, our aim is to *approximate* the probability that a parameter is outside a region of practical importance (or that the absolute value of the parameter exceeds a threshold). MCMC methods can certainly be used for this purpose but our intent is to provide a simpler, quicker, and less computationally intensive approach that can be utilized as part of the model evaluation process for local fit assessment. The complex posterior distribution is simplified as we shift from sampling from the marginal posterior to a specific conditional posterior for each parameter. That is, observations are sampled from each posterior by fixing estimates of loadings, error (co)variances, factor (co)variances, etc. in order to reduce the model down to a single parameter problem. Then, Laplace's method is employed to form a normal distribution approximation to the conditional posterior of parameter(s) of interest which we can then draw samples to from the approximated posterior. The steps of the proposed algorithm are described in Appendix A and the R code used for this algorithm is shown in Appendix B.

Illustrative Example of Local Fit Assessment with Simulated Data

To illustrate the proposed method, consider the "true" factor model shown in Figure 1. The simple three factor structure was selected for illustrative purposes. Using a sample of 500 drawn from this model, a similar, yet misspecified model ignoring the residual covariances was estimated using lavaan (Rosseel, 2012). Based on the model χ^2, the model does not fit these data ($\chi^2(51) = 124.3, p < 0.001$). Next,one must attempt to diagnose the source of inadequate model-data fit, such as which paths were not specified. Next, we applied the described probabilistic approach to try to identify omitted paths that may be

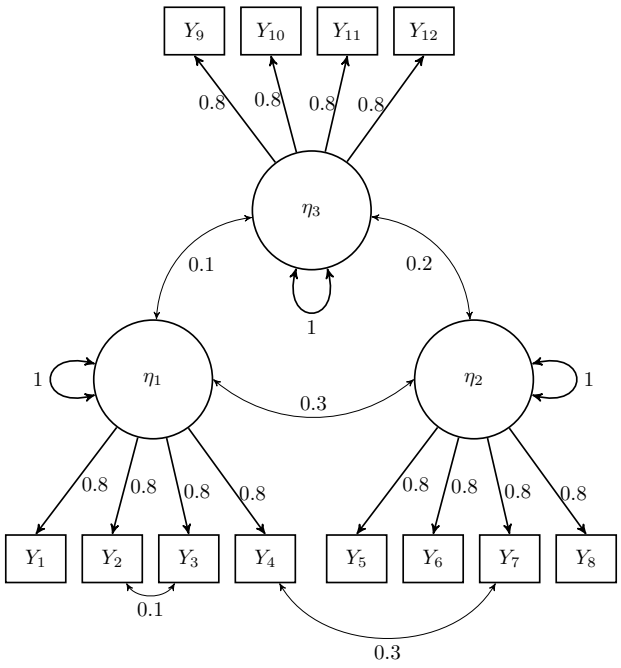

Figure 1
Data generating model

of substantive interest. The results from the proposed local fit assessment are shown in Table 1. The paths (i.e., loading or error covariances) identified as most likely of substantive interest are shown in descending order. For simplicity we are only showing the paths with the highest nine probabilities plus the covariance for Y_2 and Y_3. These estimates are shown in Table 1. We found that, with the method described in this paper, the path for the covariance between Y_4 and Y_7 had the highest probability of being meaningful in magnitude whereas the covariance between Y_2 and Y_3 was unlikely to be meaning. This matches well with the true model where the covariance between Y_4 and Y_7 was 0.30 which is above the cutoff of 0.25 whereas the covariance between Y_2 and Y_3 was 0.10 in the population which is below the cutoff.

Simulation Study to Evaluate Posterior Probability

In a small scale simulation study, we evaluated the estimated probabilities against the sampling distribution of the parameters. To do so, we generated 10,000 datasets from the population model, fit the population model to each, and extracted the sampling

Table 1
Estimated probabilities of parameters being outside region of practical equivalence

θ	$\Pr(\mid \theta \mid \geq \text{cutoff})$
$\sigma_{7,4}$	**.998**
$\lambda_{7,1}$	.014
$\sigma_{5,1}$	.012
$\sigma_{11,4}$	.009
$\lambda_{6,1}$	.005
$\sigma_{9,7}$	.004
$\sigma_{3,2}$	**.001**

Note. Bolded values are for parameters that are from the true generating model. Cutoff for σ was 0.25 and cutoff for λ was 0.30.

distribution of the model parameters. We compared the sampling distribution to the approximated empirical sampling distributions used to compute the approximating probabilities shown in Table 1. We focused on the two residual covariances from the model in Figure 1, because these two parameters are likely to be initially omitted from a model specification. The true and approximated sampling distributions are shown in Figure 2. The probabilities from the "true" sampling distribution are $\Pr(\sigma_{7,4} > 0.25) = 0.83$ and $\Pr(\sigma_{3,2} > 0.25) = 0.01$.

Discussion

In this paper, we have outlined a new method for investigating local fit by approximating the probability that a parameter is meaningfully different than zero, which builds on the previous work of Lee et al. (2016) and Shi et al. (2019). By utilizing a Bayesian perspective, the resulting probability is arguably more intuitive to interpret. The availability of a local fit assessment that does not require a complex interpretation may complement existing methods of model modification (e.g., Kaplan, 1989; Sörbom, 1989;

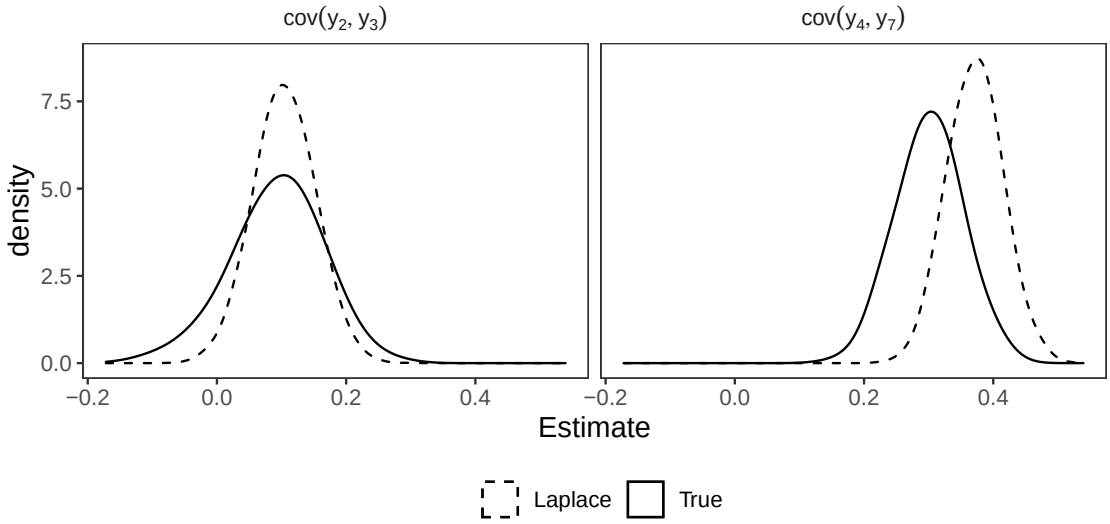

Figure 2
True and approximated sampling distributions for model residual covariances

Wald, 1943). By using a straightforward approximation based on Laplace's method, many estimation complexities are avoided. The approximation is also relatively fast compared to Markov chain Monte Carlo methods.

Despite some potential advantages of the proposed method, there are also some important considerations. The first consideration is determining the appropriate value of each parameter that should be used for computing the probability. For (standardized) factor loadings, we used a value of 0.30 suggested by Benson and Nasser (1998) from EFA, but other values may be of interest. Many of the decisions required for this method are likely to depend on the inferences of interest. For example, the threshold of meaningful correlation values will vary by application. Furthermore, researchers must decide the probability threshold needed in order to consider an omitted path meaningful. A fixed value for this probability may be counter productive because this probability is meant to be reflective of degree of evidence or belief about the magnitude of the parameter. This view of probability is sometimes called the *epistemic* or *degree-of-belief* perspective of probability (de Finetti, 1974; Levy & Mislevy, 2016, p. 14-15).

In conclusion, we have provided an approach to local fit assessment that provides

researchers for an interpretable estimate of whether a modification will be of substantive importance. The proposed approach allows for more information to be gained about the relationships in an observed dataset with respect to one's inferential interests. This proposed approach is arguably intuitive and aligns with aims to search for sources of local (mis)fit rather than global fit.

References

Benson, J., & Nasser, F. (1998). On the Use of Factor Analysis as a Research Tool. *Journal of Vocational Education Research*, *23*(1), 13–33.

Bentler, P. M. (1990). Comparative fit indexes in structural models. *Psychological Bulletin*, *107*(2), 238–246. https://doi.org/10.1037/0033-2909.107.2.238

Bentler, P. M. (1995). *EQS structural equations program manual*. Multivariate Software.

Bollen, K. A. (1995). Structural equation models that are nonlinear in latent variables: A least-squares estimator. *Sociological Methodology*, *25*, 223. https://doi.org/10.2307/271068

Bollen, K. A. (2019). Model Implied Instrumental Variables (MIIVs): An Alternative Orientation to Structural Equation Modeling. *Multivariate Behavioral Research*, *54*(1), 31–46. https://doi.org/10.1080/00273171.2018.1483224

Browne, M. W., & Cudeck, R. (1992). Alternative Ways of Assessing Model Fit. *Sociological Methods & Research*, *21*(2), 230–258. https://doi.org/10.1177/0049124192021002005

Buse, A. (1982). The Likelihood Ratio, Wald, and Lagrange Multiplier Tests: An Expository Note. *The American Statistician*, *36*(3a), 153–157. https://doi.org/10.1080/00031305.1982.10482817

Carpenter, B., Gelman, A., Hoffman, M. D., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M., Guo, J., Li, P., & Riddell, A. (2017). Stan: A probabilistic programming language. *Journal of statistical software*, *76*(1). https://doi.org/10.18637/jss.v076.i01

Chou, C. P., & Bentler, P. M. (1990). Model Modification in Covariance Structure Modeling: A Comparison among Likelihood Ratio, Lagrange Multiplier, and Wald Tests. *Multivariate Behavioral Research*, *25*(1), 115–136. https://doi.org/10.1207/s15327906mbr2501_13

de Finetti, B. (1974). *Theory of probability, violumne 1*. Wiley.

DiStefano, C. (2016). Examining Fit With Structural Equation Models. In K. Schweizer & C. DiStefano (Eds.), *Principles and methods of test construction* (pp. 166–193). Hogrefe Publishing.

Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., & Rubin, D. B. (2013). *Bayesian data analysis* (3rd). CRC press.

Gelman, A., Meng, X. L., & Stern, H. (1996). Posterior predictive assessment of model fitness via realized discrepancies. *Statistica Sinica*, *6*(4), 733–807.

Jöreskog, K. G., & Sörbom, D. (1981). *LISREL V: Analysis of linear structural relationships by the method of maximum likelihood*. National Educational Resources Jöreskog, K. G., & Sörbom, D. (1981). USREL V: Analysis of linear structural relationships by the method of maximum likelihood. Chicago: National Educational Resources.

Kaplan, D. (1989). Model Modification in Covariance Structure Analysis: Application of the Expected Parameter Change Statistic. *Multivariate Behavioral Research*, *24*(3), 285–305. https://doi.org/10.1207/s15327906mbr2403_2

Kass, R. E., & Steffey, D. (1989). Approximate bayesian inference in conditionally independent hierarchical models (Parametric empirical bayes models). *Journal of the American Statistical Association*, *84*(407), 717–726. https://doi.org/10.1080/01621459.1989.10478825

Kline, R. B. (2015). *Principles and practice of structural equation modeling* (Fourth). The Guilford Press.

Lee, T., Cai, L., & Kuhfeld, M. (2016). A poor person's posterior predictive checking of structural equation models. *Structural Equation Modeling*, *23*(2), 206–220. https://doi.org/10.1080/10705511.2015.1014041

Levy, R., & Choi, J. (2013). Bayesian structural equation modeling. In G. R. Hancock & R. O. Mueller (Eds.), *Structural equation modeling: A second course* (pp. 563–623). Information Age Publishing.

Levy, R., & Mislevy, R. J. (2016). *Bayesian psychometric modeling*. CRC Press.

Li, B. (1992). Laplace expansion for posterior densities of nonlinear functions of parameters. *Biometrika*, *79*(2), 393–401.

Maydeu-Olivares, A., & Shi, D. (2017). Effect sizes of model misfit in structural equation models: Standardized residual covariances and residual correlations. *Methodology*, *13*(Supplement 1), 23–30. https://doi.org/10.1027/1614-2241/a000129

Maydeu-Olivares, A., Shi, D., & Rosseel, Y. (2018). Assessing Fit in Structural Equation Models: A Monte-Carlo Evaluation of RMSEA Versus SRMR Confidence Intervals and Tests of Close Fit. *Structural Equation Modeling*, *25*(3), 389–402. https://doi.org/10.1080/10705511.2017.1389611

Merkle, E. C., & Rosseel, Y. (2018). blavaan: Bayesian structural equation models via parameter expansion. *Journal of Statistical Software*, *85*(4), 1–30. https://doi.org/10.18637/jss.v085.i04

Muthén, L., & Muthén, B. (2017). *Mplus User's Guide* (8th). Muthén & Muthén.

Neyman, J., & Pearson, E. S. (1928). On the use and interpretation of certain test criteria for purposes of statistical inference. *Biometrika*, *20*(3/4), 175–240, 263–294. https://doi.org/10.2307/2332112

Plummer, M. et al. (2003). Jags: A program for analysis of bayesian graphical models using gibbs sampling. *Proceedings of the 3rd international workshop on distributed statistical computing*, *124*(125.10), 1–10.

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

Rubin, D. B. (1996). Comment: On posterior predictive p-values. *Statistica Sinica*, *6*(4), 787–792.

Rue, H., Martino, S., & Chopin, N. (2009). Approximate Bayesian inference for latent Gaussian models by using integrated nested Laplace approximations. *Journal of the*

Royal Statistical Society. Series B: Statistical Methodology, *71*(2), 319–392. https://doi.org/10.1111/j.1467-9868.2008.00700.x

Shi, D., Song, H., DiStefano, C., Maydeu-Olivares, A., McDaniel, H. L., & Jiang, Z. (2019). Evaluating Factorial Invariance: An Interval Estimation Approach Using Bayesian Structural Equation Modeling. *Multivariate Behavioral Research*, *54*(2), 224–245. https://doi.org/10.1080/00273171.2018.1514484

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

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/10.1016/j.paid.2006.09.017

Surhone, L. M., Tennoe, M. T., & Henssonow, S. F. (2010). *Openbugs*. Betascript Publishing. https://doi.org/10.5555/1941071

Tierney, L., & Kadane, J. B. (1986). Accurate approximations for posterior moments and marginal densities. *Journal of the American Statistical Association*, *81*(393), 82–86. https://doi.org/10.1080/01621459.1986.10478240

Wald, A. (1943). Tests of Statistical Hypotheses Concerning Several Parameters When the Number of Observations is Large. *Transactions of the American Mathematical Society*, *54*(3), 426. https://doi.org/10.2307/1990256

Whittaker, T. A. (2012). Using the Modification Index and Standardized Expected Parameter Change for Model Modification. *The Journal of Experimental Education*, *80*(1), 26–44. https://doi.org/10.1080/00220973.2010.531299

Wolfinger, R. (1993). Laplace's approximation for nonlinear mixed models. *Biometrika*, *80*(4), 791–796.

Appendix A

Laplace Approximation Algorithm Steps

Consider the model, Equation 1, with corresponding full posterior in Equation 2. Laplace's method is utilized by constructing individual single parameter posteriors and sampling from each posterior separately. This approach can be achieved through the following steps.

1. Fit the initially proposed CFA model (M) and save all the parameter estimates (θ).

2. Construct a list of all possible parameters in M that are not currently in the parameter space, such as all cross-loadings and error-covariances fixed to zero. Call these parameters θ^0.

3. Fix the parameters in M to those initially estimates, θ. For each identified parameter in θ^0, optimize the log posterior with respect θ^0. This step with result in a posterior expected value for θ^0 and the variance of this estimate.

4. Use the estimated expected value and variance as the parameters in a normal approximation to the posterior distribution. Sample N times from this posterior and save these draws.

5. After all draws for each parameter are saved, standardized each draw based on the model estimated variances from M.

6. For each parameter, loop over all draws from the posterior to identify whether the sample exceeded the cutoff for the practical significance. Then, compute the mean of the 0/1 vector for each parameter. This mean is the posterior probability that the parameter is practically significant.

Appendix B

R Code for Laplace Approximation

```
# ========================================== #
# ========================================== #
#   function: convert2matrix()
# ========================================== #
# use: takes vectors of positions in matrix
#       and values to create a possible
#       sparse matrix
#
# arguments:
# theta - vector of parameters being optimized
# X     - obversed data
# model - list of model components (lambda...)
#
convert2matrix <- function(x, y, est) {
  Z<- matrix(NA, nrow=max(x), ncol=max(y))
  for(i in 1:length(est)){
    Z[x[i], y[i]] <- est[i]
  }
  Z
}

# ========================================== #
# ========================================== #
#   function: get_starting_values()
# ========================================== #
# use: obtain starting values, will work
#       for any parameters
#
# arguments:
# model - list of model components (lambda...)
#
get_starting_values <- function(model){
  lambdaMod <- model[[1]]
  phiMod <- model[[2]]
  psiMod <- model[[3]]

  # get length of each model element
  lam.num <- sum(is.na(c(lambdaMod))==T)
  phi.num <- sum(is.na(c(phiMod))==T)
  dphi.num <- sum(is.na(diag(phiMod))==T)
  odphi.num <- sum(is.na(phiMod[lower.tri(phiMod)])==T)
  psi.num <- sum(is.na(c(psiMod))==T)
  dpsi.num <- sum(is.na(diag(psiMod))==T)
```

```
odpsi.num <- sum(is.na(psiMod[lower.tri(psiMod)])==T)
# tau.num <- p
# eta.num <- length(etaMod)

k<-lam.num+phi.num+psi.num#+tau.num+eta.num
sv<-numeric(k)
# generate starting values
sv.n1 <- sv.n2 <- sv.n3 <- sv.n4 <- sv.n5 <- NA
if(lam.num==0){
  sv.n1 <- NA
}else{
  sv[1:(lam.num)] <- runif(lam.num, 0.6, 0.8)
  sv.n1 <-   paste0('lambda', 1:lam.num)
}
if(dphi.num==0){
  sv.n2 <- NA
}else{
  sv[(lam.num+1):(lam.num+dphi.num)]<- runif(dphi.num, 0.05, 1)
  sv.n2 <-   paste0('dphi', 1:dphi.num)
}
if(odphi.num==0){
  sv.n3 <- NA
}else{
  sv[(lam.num+dphi.num+1):(lam.num+phi.num)]<- runif(odphi.num,
     -.1, 0.1)
  sv.n3 <-  paste0('odphi', 1:odphi.num)
}

if(dpsi.num==0){
  sv.n4 <- NA
}else{
  sv[(lam.num+phi.num+1):(lam.num+phi.num+dpsi.num)] <- rep(0.2,
     dpsi.num)
  sv.n4 <-  paste0('dpsi', 1:dpsi.num)
}

if(odpsi.num==0){
  sv.n5 <- NA
}else{
  sv[(lam.num+(phi.num + dpsi.num)+1):(lam.num+phi.num+psi.num)]
     <- runif(odpsi.num, -.05, 0.05)
  sv.n5 <-  paste0('odpsi', 1:odpsi.num)
}
#sv[(lam.num+phi.num+psi.num+1):(lam.num+phi.num+psi.num+tau.num)]
    <-sample(unlist(X), tau.num, replace=T)
#sv[(lam.num+phi.num+psi.num+tau.num+1):(lam.num+phi.num+psi.num+
   tau.num+eta.num)] <-rnorm(eta.num, 0, 1)
```

```
  names(sv) <- na.omit(c(sv.n1, sv.n2, sv.n3, sv.n4, sv.n5)) #,
     paste0('tau', 1:tau.num), paste0('eta', 1:eta.num))
  return(sv)
}
# ========================================== #
# ========================================== #
#   function: trace()
# ========================================== #
# use: compute trace of matrix
#
trace <- function(A) {
  n <- dim(A)[1] # get dimension of matrix
  tr <- 0 # initialize trace value

  # Loop over the diagonal elements of the supplied matrix and add
     the element to tr
  for (k in 1:n) {
    l <- A[k,k]
    tr <- tr + l
  }
  return(tr[[1]])
}
# or one could do sum(diag(A))
# ========================================== #
# ========================================== #
#   function: XX2full()
# ========================================== #
# use: convert lower or upper matrix to full
up2full <- function(m) {
  m[lower.tri(m)] <- t(m)[lower.tri(m)]
  m
}
low2full <- function(m) {
  m[upper.tri(m)] <- t(m)[upper.tri(m)]
  m
}

# ========================================== #
# ========================================== #
#   function: progress_bar()
# ========================================== #
# use: makes the progress bar
progress_bar <- txtProgressBar(min = 0, max = 2, style = 3)

# ========================================== #
# ========================================== #
#   function: get_prior_dens()
```

```
# ========================================== #
# use: gets the appropriate prior for the
#       parameter of interest
#
get_prior_dens <- function(pvalue, pname,...){
  if(pname %like% 'lambda'){
    out <- dnorm(pvalue, 0, 1, log=T)
  }
  if(pname %like% 'dphi'){
    out <- dgamma(pvalue, 1, 0.5, log=T)
  }
  if(pname %like% 'odphi'){
    out <- dnorm(pvalue, 0, 1, log=T)
  }
  if(pname %like% 'dpsi'){
    out <- dgamma(pvalue, 1, 0.5, log=T)
  }
  if(pname %like% 'odpsi'){
    out <- dnorm(pvalue, 0, 1, log=T)
  }
  if(pname %like% 'eta'){
    out <- dnorm(pvalue, 0, 10, log=T)
  }
  if(pname %like% 'tau'){
    out <- dnorm(pvalue, 0, 32, log=T)
  }
  return(out)
}

# ========================================== #
# ========================================== #
#   function: get_log_post()
# ========================================== #
# use: uses the model, parameters, and data to
#       to calculate log posterior
#
# arguments:
# p        - names vector of parameters
# sample.data - data frame of raw data
# cfa.model - list of model components
#
get_log_post <- function(p, sample.data, cfa.model,...) {

  out <- use_cfa_model(p, cov(sample.data), cfa.model)

  log_lik <- sum(apply(sample.data, 1, dmvnorm,
                       mean=out[['tau']],
```

```
                        sigma=out[['Sigma']], log=T))

  log_prior<-0
  if(length(p)==1){
    log_prior <- get_prior_dens(p, names(p))
  } else {
    i <- 1
    for(i in 1:length(p)){
      log_prior <- log_prior + get_prior_dens(p[i], names(p)[i])
    }
  }
  log_post <- log_lik + log_prior
  log_post
}

# ========================================== #
# ========================================== #
#   function: use_cfa_model()
# ========================================== #
# use: take in parameters, data, and model to
#         obtain the log-likelihood
#
# arguments:
# theta - vector of parameters being optimized
# sample.cov - samplecovariance matrix
# cfa.model - list of model parameters
use_cfa_model <- function(theta, sample.cov, cfa.model,...){
  # Compue sample statistics
  p<-ncol(sample.cov)
  S<-sample.cov

  # unpack model
  lambda <- cfa.model[[1]]
  phi <- cfa.model[[2]]
  psi <- cfa.model[[3]]
  #tau <- cfaModel[[4]]
  #eta <- cfaModel[[5]]

  # number factor loadings
  lam.num <- length(which(is.na(lambda)))
  lambda[which(is.na(lambda))] <- theta[1:lam.num]
  nF = ncol(lambda)
  # number elements in factor (co)variance matrix
  phi.num <- length(which(is.na(phi)))
  dphi.num <- sum(is.na(diag(phi))==T)
  odphi.num <- sum(is.na(phi[lower.tri(phi)])==T)
  if(phi.num > 0){
```

```
  if(dphi.num == 0){
    phi[which(is.na(phi))] <- theta[(lam.num+1):(lam.num+phi.num)]
  } else {
    diag(phi) <- theta[(lam.num+1):(lam.num+dphi.num)]
    phi[which(is.na(phi))] <- theta[(lam.num+dphi.num+1):(lam.num+
       phi.num)]
  }
}
phi <- low2full(phi) # map lower to upper

# number elements in error (co)variance matrix
psi.num <- length(which(is.na(psi)))
dpsi.num <- sum(is.na(diag(psi))==T)
odpsi.num <- sum(is.na(psi[lower.tri(psi)])==T)
if(psi.num > 0){
  if(dpsi.num == 0){
    psi[which(is.na(psi))] <- theta[(lam.num+1):(lam.num+psi.num)]
  } else {
    diag(psi) <- theta[(lam.num+1):(lam.num+dpsi.num)]
    psi[which(is.na(psi))] <- theta[(lam.num+dpsi.num+1):(lam.num+
       psi.num)]
  }
}
psi <- low2full(psi)
# number of factor scores
#eta.num <- length(eta)
#eta <- matrix(theta[(lam.num+phi.num+psi.num+tau.num+1):(lam.num+
   phi.num+psi.num+tau.num+eta.num)],
#              nrow=nF)
# mean center eta
#for(i in 1:nF){
#  eta[i, ] <- eta[i,] - mean(eta[,i])
#}

# # number of intercepts
# tau.num <- length(tau)
# tau <- matrix(theta[(lam.num+phi.num+psi.num+1):(lam.num+phi.num
   +psi.num+tau.num)], ncol=1)
# tau <- repeat_col(tau, ncol(eta))

# compute model observed outcomes
#Y <- tau + lambda%*%eta
tau <- numeric(p)
# compute model implied (co)variance matrix
Sigma<-lambda%*%phi%*%(t(lambda)) + psi

#return fit value
```

```
  out <- list(Sigma, lambda, phi, psi, tau)
  names(out) <- c('Sigma', 'lambda', 'phi', 'psi', 'tau')
  return(out)
}



# ========================================== #
# ========================================== #
#   function: laplace_local_fit()
# ========================================== #
# use: uses the fittes lavaan object to run
#       the proposed method
#
# arguments:
# fit       - fitted lavaan model
# standardized - logical for whether to standardize
# cut.load  - cutoff for value of loading to care about default =
   0.3
# cut.cov   - cutoff for value of covariances to care about default
   = 0.1
# opt       - list of parameters to pass to interior functions
# sum.print - logical indicator of whether to print the summary
   table upon completion
# counter   - logical indicator of whether to print out a (.) after
   each
#               parameter is completed
#
laplace_local_fit <- function(fit, cut.load = 0.3, cut.cov = 0.1,
   standardize=T,
                              opt=list(scale.cov=1, no.samples=1000)
                                 ,
                              all.parameters=F,
                              sum.print=F, pb=T,...){

  # Observed Data
  sampleData <- fit@Data@X[[1]]
  # sample covariance matrix
  sampleCov <- fit@SampleStats@cov[[1]]

  # extract model
  extractedLavaan <- lavMatrixRepresentation(partable(fit))

  factNames <- unique(extractedLavaan[extractedLavaan[,"mat"]=="
     lambda", "lhs"])
  varNames <- unique(extractedLavaan[extractedLavaan[,"mat"]=="
     lambda", "rhs"])
```

```
# extract factor loading matrix
lambda <- extractedLavaan[ extractedLavaan$mat == "lambda" ,]
lambda <- convert2matrix(lambda$row, lambda$col, lambda$est)
colnames(lambda) <- factNames
rownames(lambda) <- varNames
# extract factor covariance matrix
phi <- extractedLavaan[ extractedLavaan$mat == "psi" ,]
phi <- convert2matrix(phi[,'row'], phi[,'col'], phi[,'est'])
phi <- up2full(phi)
colnames(phi) <- rownames(phi) <- factNames
# extract error covariance matrix
psi <- extractedLavaan[ extractedLavaan$mat == "theta" ,]
psi <- convert2matrix(psi[,'row'], psi[,'col'], psi[,'est'])
psi[upper.tri(psi)] <- 0
colnames(psi) <- rownames(psi) <- varNames


# need to create list of all NA parameters in the above matrices

if(all.parameters == T){
  lambdaA <- lambda
  phiA <- phi
  psiA <- psi

  lambdaA[!is.na(lambdaA)] <- NA
  phiA[!is.na(phiA)] <- NA
  psiA[!is.na(psiA)] <- NA

} else{
  lambdaA <- lambda
  phiA <- phi
  psiA <- psi

}

lamList <- as.matrix(which(is.na(lambdaA), arr.ind = T))
il <- nrow(lamList)
phiList <- as.matrix(which(is.na(phiA), arr.ind = T))
ip <- il + nrow(phiList)
psiList <- as.matrix(which(is.na(psiA), arr.ind = T))
it <- ip + nrow(psiList)
modList <- rbind(lamList, phiList, psiList)
# number of variables
# create names for each condition
vnlamList <- lamList
vnlamList[,2] <- paste0(factor(vnlamList[,2], levels = order(
   unique(vnlamList[,2])),labels=factNames))
```

```
vnlamList[,1] <- rownames(lamList)
vnlamList[,2] <- paste0(vnlamList[,2],"=~",vnlamList[,1])
vnphiList <- phiList
if(nrow(phiList)>0){
  vnphiList[,1] <- paste0(factor(phiList[,1], levels = order(
     unique(vnphiList[,1])),labels=factNames))
  vnphiList[,2] <- paste0(factor(phiList[,2], levels = order(
     unique(phiList[,2])),labels=factNames))
}
vnpsiList <- psiList
vnpsiList[,1] <- rownames(psiList)
vnpsiList[,2] <- paste0(vnpsiList[,1],"~~y", psiList[,2])
nameList <- rbind(vnlamList, vnphiList, vnpsiList)
# ========================================================== #
# ========================================================== #
# iterate around this function
fitResults <- matrix(nrow=opt[[2]], ncol=it)
# progress bar
if(pb==T) progress_bar <- txtProgressBar(min = 0, max = it, style
   = 3)
iter <- 1
for(iter in 1:it){

  # extract iteration information from modList
  x <- modList[iter, ]

  # do we need to update lambda?
  if(iter <= il){
    Q <- lambda
    Q[is.na(Q)] <- 0
    Q[x[1], x[2]] <- NA
    lambdaMod <- Q
  } else {
    Q <- lambda
    Q[is.na(Q)] <- 0
    lambdaMod <- Q
  }

  # update phi?
  if(iter > il & iter <= ip){
    Q <- phi
    Q[is.na(Q)] <- 0
    Q[x[1], x[2]] <- NA
    phiMod <- Q
  } else {
    Q <- phi
    Q[is.na(Q)] <- 0
```

```
    phiMod <- Q
  }

  # update psi?
  if(iter > ip){
    Q <- psi
    Q[is.na(Q)] <- 0
    Q[x[1], x[2]] <- NA
    psiMod <- Q
  } else {
    Q <- psi
    Q[is.na(Q)] <- 0
    psiMod <- Q
  }

  # combine into a single list
  cfaModel <- list(lambdaMod, phiMod, psiMod) #, tauMod, etaMod

  #print(cfaModel)
  # get starting values
  inits <- get_starting_values(cfaModel)

  # use optim() to run simulation
  fit <- optim(inits, get_log_post, control = list(fnscale = -1),
               hessian = TRUE,
               sample.data=sampleData, cfa.model=cfaModel)
  param_mean <- fit$par # numerical deriv
  # compute hess at param_mean
  #hess <- numDeriv::hessian(model, param_mean, ...)
  #param_cov_mat <- solve(-hess)
  param_cov_mat <- solve(-fit$hessian)

  # scaled covariance matrix (artifically inflate uncertainty)
  scale.cov = opt[[1]]
  A <- diag(scale.cov, nrow=nrow(param_cov_mat), ncol=ncol(param_
     cov_mat))
  param_cov_mat <- A%*%param_cov_mat%*%t(A)

  # sample
  no.samples=opt[[2]]
  fitResults[,iter] <- mcmc(rmvnorm(no.samples, param_mean, param_
     cov_mat))

  if(pb == T) setTxtProgressBar(progress_bar, iter)
}
# ========================================================== #
# ========================================================== #
```

```
colnames(fitResults) <- nameList[,2, drop=T]

# Next, standardized (if desired) default
if(standardize==T){
  # standardize
  obs.var <- extractedLavaan[extractedLavaan[,"mat"]=="theta", ]
  obs.var <- obs.var[which(obs.var$lhs == obs.var$rhs), c("lhs", "
     est")]

  fct.var <- extractedLavaan[extractedLavaan[,"mat"]=="psi", ]
  fct.var <- fct.var[which(fct.var$lhs == fct.var$rhs), c("lhs", "
     est")]

  all.var <- rbind(obs.var, fct.var)

  fitResults <- fitResults
  p <- colnames(fitResults)
  i <- 1
  for(i in 1:length(p)){
    unstd <- fitResults[,i]

    if(p[i] %like% "=~"){
      pp <- strsplit(p[i], "=~") %>% unlist()
      sigjj <- sqrt(all.var[all.var[,1] == pp[1], 2])
      sigii <- sqrt(all.var[all.var[,1] == pp[2], 2])
      std <- unstd*sqrt(sigjj/sigii) # bollen (1989, p. 349)
    }

    if(p[i] %like% "~~"){
      pp <- strsplit(p[i], "~~") %>% unlist()
      sigjj <- sqrt(all.var[all.var[,1] == pp[1], 2])
      sigii <- sqrt(all.var[all.var[,1] == pp[2], 2])
      std <- unstd/(sigjj * sigii) # bollen (1989, p. 349)
    }

    fitResults[,i] <- std
  }
}
# now, compute and format summary statistics
sumResults <- data.frame(matrix(nrow=ncol(fitResults), ncol=9))
colnames(sumResults) <- c("Parameter","Prob", "mean", "sd", "p0
   .025", "p0.25", "p0.5", "p0.75", "p0.975")
sumResults[,1] <- colnames(fitResults)

sumResults[,3:9] <- t(apply(fitResults, 2, function(x){
  c(mean(x, na.rm=T), sd(x, na.rm=T),
```

```
      quantile(x, c(0.025, 0.25, 0.5, 0.75, 0.975), na.rm=T))
  }))

  # compute probability of meaningfulness
  # depends on parameter
  # cut.load = 0.3
  # cut.cov = 0.1
  p <- colnames(fitResults)
  for(i in 1:ncol(fitResults)){
    x <- fitResults[,i, drop=T]
    if(p[i] %like% "=~"){
      pv <- mean(ifelse(abs(x) >= cut.load, 1, 0))
    }
    if(p[i] %like% "~~"){
      pv <- mean(ifelse(abs(x) >= cut.cov, 1, 0))
    }
    sumResults[i, 2] <- pv
  }
  sumResults <- arrange(sumResults, desc(Prob))
  colnames(sumResults) <- c("Parameter","Pr(|theta|>cutoff)", "mean"
     , "sd", "p0.025", "p0.25", "p0.5", "p0.75", "p0.975")
  sumResults[,2:9] <- round(sumResults[,2:9], 3)
  cat("\n")
  if(sum.print==T) print(sumResults, row.names = FALSE)

  # convert to data.frame
  fitResults <- as.data.frame(fitResults)
  out <- list(fitResults, sumResults)
  names(out) <- c("All Results", "Summary")

  return(out)
}
```