Research Article

Open Access CrossMark

Estimating structural equation models within a bayesian framework: a concrete example of a markov chain monte carlo method

Abstract

Bayesian estimation approaches can accommodate a variety of latent-variable models including structural equation modeling (SEM). Despite their theoretical appeal, it can be non-trivial to implement appropriate estimation algorithms in these contexts. There is a strong need of detailed explanations of the impact of the various estimation options on how a MCMC sampler behave to empower users of Bayesian estimation methods to understand choices made by programs and, perhaps, to make them aware of challenges in developing their own MCMC estimation procedure. In this paper, a Markov chain Monte Carlo (MCMC) estimation method is illustrated using a one-factor model and a two-factor model with a directional path. The objective is to illustrate in an accessible manner how different estimation choices such as the choice of starting values, the choice of jump sizes for the proposal distribution, and the choice of model complexity impact MCMC estimation results.

Keywords: structural equation modeling, markov chain monte carlo, parameter estimation, bayesian inference, metropolis algorithm

Volume 7 Issue 1 - 2018

Jaehwa Choi
The George Washington University, USA

Correspondence: Jaehwa Choi, Associate Professor and Director, Assessment, Testing, and Measurement, The George Washington University, USA, Email jaechoi@gwu.edu

Received: December 20, 2017 | **Published:** February 15, 2018

Structural equation modeling (SEM; Bollen,[1]) is a key latent-variable modeling framework in the social and behavioral sciences. It is generally used for understanding how observable indicators relate to latent variables that are postulated to represent unobservable constructs and how these latent variables influence each other. SEM can accommodate a variety of research questions including those that lead to multi-group estimation, multi-level estimation, missing data estimation, and longitudinal estimation, both in isolation and in combination.

Mathematically, the aim of SEM is to test the hypothesis that the observed covariance matrix for a set of observable indicator variables in the population, Σ, is equal to the covariance matrix that is implied by the particular SEM that is fitted to population data, $\Sigma(\theta)$. Since population data are generally not available to researchers, this hypothesis is tested by comparing the fit of a model-implied covariance matrix to the covariance matrix computed based on sample data, $\mathbf{S}$. In the notation $\Sigma(\theta)$, θ denotes a joint vector of different types of model parameters, which typically include (a) *loadings* of observable indicator variables on the latent variables, (b) *error variances* associated with the observable indicator variables, (c) *variances* of the latent variables themselves, (d) *correlations or regression coefficients* between latent variables, and (e) *means* of latent variables. As we shall see below for the models used in this paper, all of these parameters, except for latent means, need to be estimated.

In order to assess whether the model-implied covariance matrix reproduces the observed-data matrix reasonably well, different *fit functions* or *target functions* need to be specified, whose definition depends on the estimation approach used. Common estimation approaches include *maximum likelihood* (ML) estimation (Bollen,[1]), *generalized least square* (GLS) estimation (Browne,[2]), and *weighted least squares* (WLS) estimation (Browne,[3]). All three estimation approaches provide consistent, efficient, and unbiased parameter estimates and asymptotic standard errors if their respective assumptions (e.g., adequate sample size, appropriate distributional forms of indicator and latent variables, and properly specified model structure) are met along with an omnibus test of model fit (Bollen,[1] for more details of these fit functions).

Even though these estimation methods have been widely used, they can be challenging to implement when the parametric complexity of a specified SEM model is high. This is the case, for example, when SEM models are fit to ordinal indicator variables, when mixture distributions are include in an SEM model, or when a multi-level SEM model is estimated. Consequently, these situations are one primary motivation for looking at alternative estimation approaches. The predominantly used alternative approach is a *Bayesian estimation* approach (e.g., Gelman;[4] Lee;[5] Choi & Levy;[6] Levy & Mislevy,[7]), which is the focus of this paper.

In Bayesian estimation approaches, point estimates of parameters their corresponding interval estimates are estimated from what is known as the *posterior distribution* of each parameter, which is also known as the *marginal density* of each parameter. It is a function of prior information about the parameter values, often provided by experts or assumed to be diffuse, and information about the parameter values contained in the data. Both types of information can be modeled using specific distributions. Specifically, the posterior distribution for each parameter is proportional to the product of the *prior distribution* and the *likelihood*, which means that posterior information about a model parameter is a statistical compromise between prior information and information contained in response data.

Despite this relatively straightforward conceptual notion, it is often impossible to analytically derive the exact form of the posterior distribution of each parameter, especially for complex models such as SEM models. Thus, one requires a mechanism for obtaining a *numerical approximation* of the posterior distribution, which is accomplished by *sampling* from it. Yet, sampling from the posterior distribution is also a challenging task because the posterior distribution of interest is usually a complex density function composed of several individual distribution functions for different parameter sets.

In recent years, *Markov chain Monte Carlo* (MCMC) estimation has been refined for this purpose and it is currently the methodological state-of-the-art for estimating various latent variable models. In a nutshell, a "Markov chain" is a statistical process that produces future outcomes that are independent of past outcomes, which enable us to sample each parameter as if they were *independent* each other although they were from a complex *joint* density. The term "Monte Carlo" essentially refers to the fact that sampling is done in a random, albeit structurally systematic, fashion. There exist a wide variety of different sampling algorithms under this heading, which have been designed to function optimally for different data structures (e.g., Junker,[8] for an overview).

Even though numerous rigorous treatments of MCMC estimation exist in the literature (e.g., Lee;[5] Chen, Choi, Stapleton, and Weiss,[9] or Choi and Levy,[6] in the context of SEM), there is a large gap between these theoretical treaties and practical implementations of it, which can be far from trivial. Most practically-oriented overview articles on Bayesian estimation (e.g., Kim and Bolt[10] in the context of item response theory models, or Lynch[11] for statistical models more generally) use very simple data sets and do not discuss the intricacies of actually implementing MCMC estimation for a specific model. Fortunately, there are programs available that handle some MCMC estimation choices automatically such as *BUGS* (Spiegelhalter, Thomas, Best, & Gilks,[12]) or AMOS 7.0 (Arbuckle,[13]).

Although the technical advantages of MCMC over traditional estimation approaches (e.g., ML estimation) have been advocated for, the details of MCMC estimation method for SEM are typically not widely understood among both methodologists and applied researchers working with SEM models. This is, in part, due to a lack of detailed illustrations of how to actually implement MCMC algorithms beyond merely asking a particular software program to estimate an SEM model. That is, even though programs such as BUGS or AMOS 7.0 make estimation more practically convenient, their availability alone has not helped users to understand how they function at a deeper level "behind the scenes".

The purpose of this paper is to provide an accessible illustration of how different estimation choices impact how a MCMC sampler behave to empower users of Bayesian estimation methods to understand choices made by programs and, perhaps, to make them aware of challenges in developing their own MCMC estimation procedure. The objective is to illustrate a MCMC algorithm behavior and not to provide a comprehensive review of various types of MCMC algorithms. Thus, two simple SEM models and one simple MCMC sampler will be used under a variety of estimation conditions, which are (a) different choices of starting values for model parameters, (b) different choices of jump sizes for the proposal distribution in the sampler, and (c) different levels of complexity of the SEM models. Specifically, the Metropolis algorithm (Metropolis & Ulam;[14] Metropolis, Rosenbluth, Rosenbluth, Teller & Teller,[15]) is utilized to estimate a one-factor model, which is akin to a confirmatory factor analysis model, and a two-factor model with a directional path between the latent variables under these different conditions.

Basic properties of estimation algorithms

Traditional estimation approaches such as ML, GLS, and WLS involve optimization routines that employ the first- and/or second-order partial derivatives of the target function, which makes them *gradient-based* search algorithms (Lee;[5] Jöreskog,[16]). Such algorithms include, for example, the Newton-Raphson or the Gauss-Newton method type algorithms (Kennedy & Gentle,[17]). For a detailed description of general gradient-based search algorithms, we recommend, for example, Süli and Mayers[18] and for a detailed descritpion gradient-based search algorithms within an SEM context we recommend, for example, Lee.[5]

For the purposes of this paper, suffice it to say that independent of which optimization algorithm is employed, the basic idea of different algorithms is the same. Using the first- and/or second-order partial derivatives, these algorithms iteratively search the parameter space of the target function to determine its minimum or maximum, depending on how the target function is defined. The maxima (or minima) found upon convergence are the ML estimates of the parameters, and their standard errors can be computed using the inverse of the Hessian matrix, which is the matrix of second-order partial derivatives.

Challenges in gradient-based estimation methods

Determining parameter estimates using gradient-based estimation is, metaphorically speaking, akin to mountain climbing such that the goal is to find the highest mountaintop among all of the mountains in a particular range. The biggest challenge is that there might be multiple mountaintops that one could aim for (i.e., multiple *local maxima* exist), and that the mountaintop that one ends up climbing depends on where one starts the climb as some mountaintops are hidden from one's view at a particular location (i.e., the *global maximum* might not be reached).

As the sample size decreases, the problem of finding local, rather than global, maxima tends to get worse. Furthermore, these estimation algorithms might not even converge to any solution when sample sizes are small (e.g., less than 100 observations) and/or when models become very complex (Browne;[3] Choi, et al.,[19] Choi, et al.,[20]).

Advantages of MCMC estimation methods

MCMC estimation methods, which are *simulation-based* estimation methods, provide a powerful, albeit not perfect, solution to these problems. Nevertheless, MCMC estimation provides a virtually universal tool to deal with optimization problems generally (Andrieu, et al.,[21] Dyer, et al.,[22] Jerrum & Sinclair,[23]). MCMC estimation approaches also have been shown to provide methodological advantages in SEM specifically. For example, they can yield reliable parameter estimates even with small sample sizes, they can flexibly accommodate data missing by design or at random, and they can handle non-linear, mixture, or hierarchical models (e.g., Arminger, & Muthén;[24] Lee;[5] Lee & Song;[25] Lee & Song;[26] Scheines, Hoijtink, & Boomsma;[27] Shi & Lee;[28] Song & Lee;[29] Song & Lee,[30]).

Importantly, the practical implementation of MCMC algorithms requires not merely generally appropriate computational strategies, but computational strategies that are specifically tailored to the structure of the SEM models for the purpose of making its estimation *efficient*. There are various decisions that need to be made that can influence the performance of an MCMC estimation approach. Some of these conditions are illustrated in this paper, but we note that other estimation aspects need to be addressed as well, which include determining (a) suitable prior distributions for parameters, (b) a sufficient number of initial samples that need to be discarded (i.e., burn-in), (c) the convergence of each parameters' posterior distribution, and (d) the number of sampling chains to run (e.g., Gilks, Richardson, & Spiegelhalter,[30]).

An overview of MCMC estimation with the metropolis algorithm

This paper focuses primarily on one sampling algorithm within the MCMC family, which is known as the *Metropolis algorithm* (Metropolis et al.,[15]). An advantage of using the Metropolis algorithm over other MCMC algorithm is that it is easy to implement. That is, its structure is fairly simple and does not involve complex numerical procedures such as adapting a secondary enveloping function, which is required in adaptive-rejection sampling (Gilks, et al.,[31]).

Likelihood function

To implement the Metropolis sampling algorithm within the SEM context, one needs to specify the likelihood function for the data, which is a distribution function for the covariance matrix of the data. A reasonable choice in this context is the *Wishart distribution*, because it has well-known properties when it is applied to variance components such as those that make up the covariance matrix. Furthermore, the well- known ML fit function is derived from this Wishart distribution (Bollen,[1]). Note, however, that the Wishart distribution is not the only option for implementing MCMC algorithms in SEM, and one advantage of the Wishart distribution based likelihood function is that it is computationally cost-effective than other types of likelihood function (e.g., Choi, et al.,[32] for more details of other likelihood function options for MCMC estimation in SEM).

As stated in the introduction, the objective of SEM is to reproduce the observed covariance matrix $\boldsymbol{S}$ as closely as possible with a particular SEM that implies a model-based covariance $\Sigma(\theta)$, which is as a function of model parameters θ. Conceptually, ML estimation treats the parameters in θ as *unknown but fixed* values; the ML estimates are those values that maximize the likelihood function. As a Bayesian method, MCMC estimation treats the parameters in θ as *random variables* and summarizes information about them using their posterior distributions. Without loosing generality, the Wishart distribution function will be denoted as $W(\mathbf{S},\Sigma(\theta))$ throughout this paper.

Overview of algorithm

The Metropolis algorithm for SEM can then be more formally described in a series of six steps that comprise *one iteration* of the algorithm:

Step 1: Establish the set of *starting values* for the parameters: $\theta^{j=0}$

Step 2: Set $j=j+1$, and draw sample a set of "candidate" values, θ^{c}, from a *proposal distribution* denoted by $q(\theta^{c}\,|\,\theta^{j-1})$

Step 3: Compute the ratio $\{\Omega(S,\Sigma(\theta^{c}))/\Omega(S,\Sigma(\theta^{j-1})$ and determine whether it is smaller or larger than 1. Choose the smaller of the two and denote it by α.

Step 4: Draw a random value u from a uniform distribution defined over (0,1).

Step 5: Compare□ α with u. If $\alpha>u$, then set $\theta^{j}=\theta^{c}$, otherwise, set $\theta^{j}=\theta^{j-1}$.

In simple terms, at each iteration, the Metropolis algorithm takes the current estimate of each parameter value and compares it to a plausible candidate, which is drawn from a so-called *proposal distribution*. If the proposed value is more likely than the current value it is chosen as the updated value for the parameter; if it is less likely, the current value is accepted with some probability.

The major computational step in the algorithm is the computation of the ratio of the two Wishart distribution values at $\theta^{c}\,and\,\theta^{j-1}$, which are denoted by $W(S,\Sigma(\theta^{c}))\,and\,W(S,\Sigma(\theta^{j-1})$, respectively. After computing this ratio, the third step involves comparing the value of the ratio to 1 and choosing the value that is smaller amongst the two. Note that the value of 1 indicates that the proposed and the current value are either equally likely (i.e., $W(S,\Sigma(\theta^{c}))=W(S,\Sigma(\theta^{j-1})$) or the proposed value is more likely than the current value (i.e., $W(S,\Sigma(\theta^{c}))>W(S,\Sigma(\theta^{j-1})$) while a value of the ratio smaller than 1 indicates that the proposed value is less likely than the current value (i.e., $W(S,\Sigma(\theta^{c}))<W(S,\Sigma(\theta^{j-1})$).

Since the Metropolis algorithm simulates a random sample from the posterior distribution under Markov chain properties, a proposed value that is less likely than a current value is not automatically discarded. Whether a proposed value that is less likely is accepted is decided by a random draw. Specifically, after randomly drawing a value u from a uniform distribution defined over the interval (0, 1) one needs to compare α and u to decide whether the proposed value should be accepted. If α is greater than the value for u, the proposed value is accepted as the jth iteration value; if α is smaller than the value for u, the current parameter value is accepted as the jth iteration value. Thus, proposed values that are *at least as likely as* current values are *always* accepted whereas proposed values that are *less likely than* current values are *sometimes* accepted in accordance with a random process. This randomness of the selection process is essential to make sure the chain (a) is not stuck on one or multiple value(s), and (b) exhaustively searches entire possible value range of the parameter. That is, it is essential to ensure that the chain converges to the target distribution of interest (Gilks, et al.,[31]). We will illustrate all these Metropolis steps with a real dataset example in the later part of this paper.

Choices in implementing MCMC estimation with the metropolis algorithm

Although the algorithm is conceptually relatively simple, however, there are three critical places within it that impact its efficiency for a particular SEM that are the focus of this paper. These are (a) choosing the starting values in step 1, (b) choosing the proposal distribution for use in step 2, and (c) choosing the length of the cycle to end after step 6 at some point in time.

Choosing appropriate starting values

Establishing *starting values* for the parameters may seem like a non-critical issue, because MCMC theory states that the distribution produced by the Markov chain will be identical to the posterior distribution of interest irrespective of the chosen starting value (Tierney,[33]). However, poorly chosen starting values may cause the algorithm to require a long burn-in period, which results in longer MCMC run times. A burn-in period is the number of samples from the estimation algorithm that need to be discarded, because the algorithm has not yet converged to the posterior distribution of interest. Therefore, even though the choice of starting values is irrelevant from a theoretical perspective, it is critical from a practical perspective. To illustrate the sensitivity of the MCMC algorithm to different starting values, ML estimates and a set of arbitrary starting values will be compared in this paper.

Choosing an appropriate proposal distribution and its parameters

Another key choice in the Metropolis algorithm is the choice of the *proposal distribution* in the second step. Interestingly, the convergence of a Markov chain toward a posterior distribution does not depend on the shape of proposal distribution in theory (Gilks, et al.,[31]). Yet, just as with the choice of starting values above, the choice of proposal distribution shape is practically important because it determines how quickly the chain converges to the posterior distribution. This had led many researchers to focus on the choice of the proposal distribution or dynamical-update adaptations of the Metropolis algorithm, which are referred to as hybrid Metropolis algorithms (Gilks, et al.,[30]).

A popular choice for the proposal distribution in general is the *normal distribution*, because (a) its characteristics are well known, (b) it matches with the target distribution which is known to be *asymptotically* (i.e., as sample size increase) normal for SEM parameters, (c) it is easy to implement for both parameters that can take on values from $-\infty \, to \, +\infty$ such as factor loadings or regression weights, and parameters that can take on only positive values such as factor or residual variances.

Formally, a normal proposal distribution q for a parameter θ in a particular iteration of the Metropolis algorithm can be expressed as $\vartheta^c \sim N(\vartheta^{j-1}, \sigma_q)$ where σ_q is the standard deviation for the proposal distribution. As this expression shows, the proposal distribution is centered at the previously accepted parameter value and σ_q governs the deviation of ϑ^c from ϑ^{j-1}. Specifically, σ_q is called the *jump size* of the proposal distribution and is selected by the analyst. The selection of σ_q plays key roles to determine how quickly the chain will converge to the target distribution, which will be illustrated in the next section of the paper.

While a normal distribution is a reasonable choice for parameters defined over the entire real number line, other distributions need to be selected of SEM parameters that are defined on more restrictive ranges of values. For example, for the purpose of ensuring positive value of factor variances and residual variances, truncated normal distributions or Gamma distributions would be more reasonable choices of proposal distribution. Specifically, the proposal distribution for the path coefficients $\vartheta = \left(b_{21}, b_{31}, b_{52}, b_{62}, \text{ and } c_{21}\right)$ in Figure 1 is set to be normal such that $\vartheta^c \sim N(\vartheta^{j-1}, \sigma_q)$ while the proposal distribution for the variance parameters

$J_1 = \left(\mathrm{Var}(E_1), \mathrm{Var}(E_2), \mathrm{Var}(E_3), \mathrm{Var}(E_4), \mathrm{Var}(E_5), \mathrm{Var}(E_6), \mathrm{Var}(F_1), \text{ and } \mathrm{Var}(D)\right)$

is set to be a truncated normal distribution on $\left[0, \infty\right)$ to constrain any parameter estimate to be positive.

While the form of the proposal distribution was not varied in this paper, the jump size (σ_q) was varied to illustrate the sensitivity of the MCMC algortihm to this implementation aspect. Specifically, the following three jump sizes were used that differed in size relative to the standard error of the ML parameter estimate, (a) small σ_q = 1/10th of the standard error of ML estimate), (b) medium (σ_q = the standard error of ML estimate), and (c) large (σ_q = 3 times of the standard error of ML estimate).

Choosing an appropriate cycle length

The third choice in the Metropolis algorithm pertains to the length of the estimation cycle. Theoretically, an MCMC estimation requires infinitely many iterations (i.e., MCMC runs or MCMC cycles) to fully converge to the posterior distribution but, of course, in practice the cycle has to be stopped at some point. The algorithm is stopped when a sample is obtained that is large enough to provide an accurate estimates (i.e., point and interval) of the parameter.

This also related to a determination of the point in the Markov chain after which it seems to behave consistently (i.e., the point at which the burn-in period ends), which can only be roughly gauged based on plots. The samples before this point are recommended to be discarded and the samples after this point are selected as samples from the posterior distribution. It is important to note, though, that there may be dependencies amongst the sampled points due to the way the algorithm functions for a particular SEM so that not every observation may be sampled in practice.

The sufficiency of the cycle length for obtaining accurate and efficient parameter estimates depends upon the complexity of the SEM. Thus, (a) a simple one-factor model with three indicators and (b) a more complex two-factor model with a directional path are estimated in this paper. The models and the data set to which they are fit are described in the following section.

Summary

Overall, this paper investigates the sensitivity of the MCMC algorithm under 2 (starting values: arbitrary, ML estimates) × 3 (jump sizes: small, medium, large) × 2 (models: simple, complex) × 1 (MCMC run: 10,000) = 12 conditions. Additionally, for the purpose of investigating the impact of MCMC run on the both point and interval estimates of model parameter, some conditions were also implemented with 1,000,000 MCMC run size. For each of these conditions, a Wishart distribution function is used for the likelihood function, a normal proposal function is used for loading parameters and the regression slope, and a truncated normal proposal function ensuring the positive values for residual and factor variances. Since this paper is not a simulation study but an illustration paper, not all possible combinations are numerically described. Instead, certain reasonable combinations are described in more detail to provide readers with a more in-depth look at how MCMC estimation with the Metropolis algorithm works in practice. For all MCMC estimations, we programmed the MCMC procedure with MATLAB 6.5 (MathWorks, Inc., [34]). The ML point estimates for starting values and standard error estimates for jump size values were obtained from EQS 6.1 (Bentler & Wu.,[35]).

Illustration

Description of sample data set

In this paper, a well-known real-data example (Reisenzein;[36] Bollen,[1]) is used to build different SEM models and to investigate the impact of different choices for starting values, jump sizes, and model complexity on the convergence of the Markov chain using the Metropolis algorithm. The context for the data is as follows. Subjects (n = 138) were asked their feeling of *sympathy* (measured by three observed variables X_1 to X_3) and *anger* (measured by three observed variables X_4 to X_6) toward another person followed by the Weiner's attribution-affect model of helping behavior (Reisenzein;[36] Weiner,[37]). The covariance matrix for these data is reproduced in Table 1.

As a result, two latent factors are specified with these indicators. The sympathy trait was represented by a one-factor model with indicators X_1 to X_3, the first indicator (X_1) is constrained to 1 for identification purposes. Similarly, the anger trait was represented by a one-factor model with indicators X_4 to X_6; the first indicator (X_4) is constrained to 1 for identification purposes as well. The two models fit in this paper are (1) the one-factor model for the sympathy trait (Model 1) and the two-factor model with for both traits with a directional path from the sympathy to the anger factor (Model 2); both models are shown graphically in Figure 1.

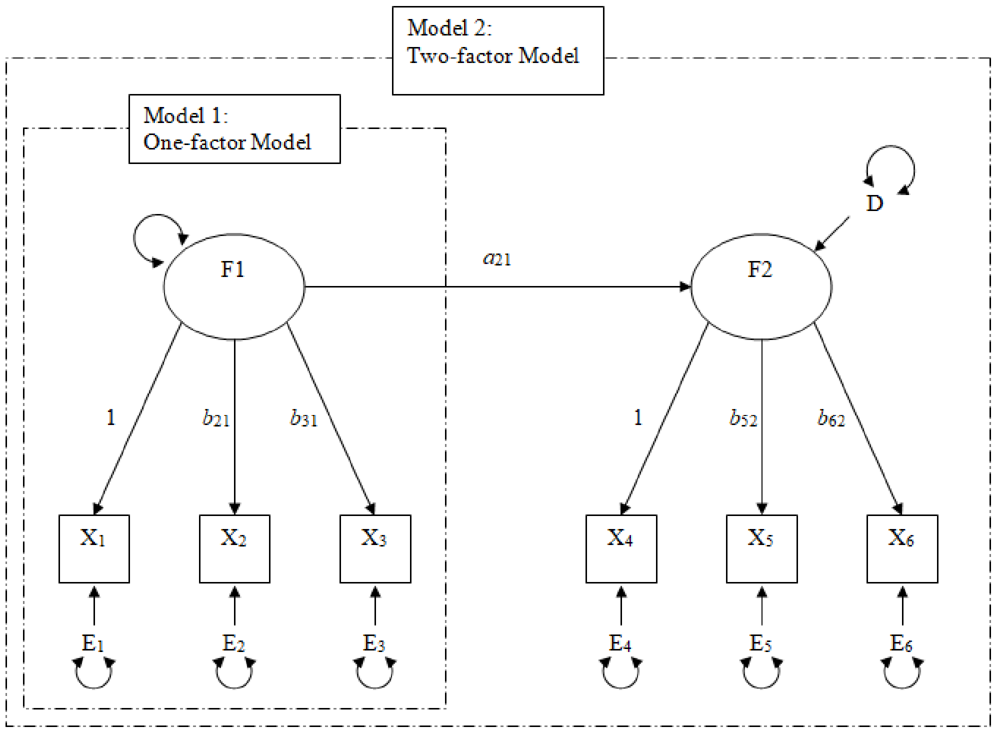

Figure 1 Diagram for the two SEM models (Model 1 and Model 2) for the sample data set analysis.

Impact summary of estimation choices

As a short summary, all different conditions along with the length of the burn-in period, the Metropolis acceptance ratio, and CPU time are shown in Table 2. Note that the Metropolis acceptance ratio is the percentage of proposed values that are accepted as random draws from the posterior distribution. An optimal range of 20-50% is recommended to achieve proper convergence (Roberts, et al.,[38] Roberts & Rosenthal,[39]).

Table 2 shows clearly that all the estimation choices that are the focus of this paper have an impact on the efficiency of the MCMC Metropolis estimation algorithm. In order to summarize the impact of the estimation choices described in the previous section on the efficiency of the Metropolis algorithm, the MCMC chain history, the Metropolis acceptance ratio, and the key characteristics of the posterior distributions of each parameter are used in the following subsections.

Choice of starting values

The specific starting values selected for both models are shown in Table 3.

Table 1 Sample covariance matrix for example data set

	X_1	X_2	X_3	X_4	X_5	X_6
X_1	6.982	4.686	4.335	-2.294	-2.209	-1.671
X_2	4.686	6.047	3.307	-1.453	-1.262	-1.401
X_3	4.335	3.307	5.037	-1.979	-1.738	-1.564
X_4	-2.294	-1.453	-1.979	5.569	3.931	3.915
X_5	-2.209	-1.262	-1.738	3.931	5.328	3.601
X_6	-1.671	-1.401	-1.564	3.915	3.601	4.977

Note Estimates are based on n = 138

Table 2 Length of burn-in, acceptance rate, and CPU time under different estimation conditions for both Model 1 and Model 2

Model number	Starting values	Jump sizes	MCMC runs	Length of burn-in	Acceptance rate	CPU time
1	Arbitrary	Small	10,000	1,000	87.81%	12.2 sec.
1	ML	Small	10,000	0	89.24%	12.1 sec.
1	Arbitrary	Medium	10,000	100	20.25%	12.3 sec.
1	ML	Medium	10,000	0	19.97%	12.3 sec.
1	Arbitrary	Large	10,000	100	1.09%	12.8 sec.
1	ML	Large	10,000	0	0.84%	12.8 sec.
2	Arbitrary	Small	10,000	> 10,000	0%	25.5 sec.
2	ML	Small	10,000	0	84.00%	25.7 sec.
2	Arbitrary	Medium	10,000	3,000	4.80%	27.2 sec.
2	ML	Medium	10,000	0	6.52%	25.1 sec.
2	Arbitrary	Large	10,000	200	0.29%	27.3 sec.
2	ML	Large	10,000	> 10,000	0%	26.4 sec.

For illustration purposes, consider estimating Model 1 under both starting value conditions with a medium jump size. Figure 2 shows the resulting chain histories and resulting posterior distributions for the model parameters.

Clearly, with arbitrary staring values the burn-in phase is approximately 100 iterations while it is much shorter under the ML starting values. That is, using ML starting values one may not need any burn-in burn in period in this case and may be able to save the entire chain history. Note that the burn-in period of 100 iterations is rather short in this example due to the simple model chosen, but the difference between the starting values holds up for more complex models as well. This can be seen in the burn-in lengths in Table 2 for Model 2, for example. We will discuss the model complexity issue more in later part of this article.

Choice of jump size for proposal distribution

For illustration purposes, consider estimating Model 1 under all three jump size conditions. Figure 3 depicts the chain history and posterior distributions for 10,000 draws for all Model 1 parameters with the small and large jump size value options using the ML starting values.

As seen in Figure 3, under the small jump size option, all parameter chains fluctuated by only small amounts throughout the 10,000 iterations, which is also known as a slow *mixing process*. Specifically, an inspection of the actual samples taken, one can observe, for example, that all first 15 proposed values were accepted. Moreover, across entire 10,000 iterations for this scenario, the Metropolis acceptance rate is high at 88.40%. In the literature, an optimal range of 20-50% is recommended to achieve proper convergence[38,39] showing that the algorithm does not work as intended in this case.

Similarly, Figure 3 shows that the chains mix poorly under a large jump size as well. An inspection of the actual samples taken shows the opposite pattern for this jump size value than what was observed above. For example, all first 15 proposed values were rejected leading to an overall Metropolis acceptance rate of only 0.048% across all 10,000 iterations. Again, this rate is far from the optimal range of 20-50% recommended in the literature. In contrast, using a medium jump size for the proposal distribution leads to a proper mixing and an overall acceptance rate of 18.59% in this scenario; Table 4 shows the first 20 iterations for this model with 5 out of the 20 values accepted.

Despite the optimal performance of the medium jump size for the proposal distribution under ML starting values compared to other conditions, one may wonder whether this poor performance cannot be compensated for by simply increasing the number of iterations. Indeed, this is the case. To illustrate the point, Table 5 shows a summary of key posterior distribution characteristics such as their mean, variance, 2.5th and 97.5th percentiles under a variety of estimation conditions.

As Table 5 makes clear, when a small jump size and arbitrary starting values – both poor estimation choices – are coupled with 1,000,000 MCMC iterations, the resulting distributional characteristics are essentially indistinguishable from when a medium jump size and ML starting values are chosen – both more optimal choices – coupled with a cycle length of only 10,000 iterations.

In other words, if one is running very long MCMC runs, the impact of starting value and/or jump size onto the marginal density estimates will diminish as the MCMC run increase. However, the 1,000,000 MCMC iterations took 21 minutes to run on a Windows machine (Intel Core 2 Duo 2.00 GHz CPU with 777 MHz 1.99 GB of RAM), which is more than 100 times longer than the 10,000 MCMC run scenario, which took only 12.5 seconds.

Impact of model complexity

Consider further that Model 1 is an extremely simple SEM model – in fact, it is the simplest possible confirmatory factor model that could be estimated – suggesting that the effects observed here will be exacerbated for more complex SEM models. To illustrate this point, consider the case when Model 2 is run under the more optimal estimation conditions of medium jump size and ML starting values. First, even though the model is only slightly more complex than Model 1 conceptually, the overall Metropolis acceptance rate is only 6.52% across 10,000 iterations under these conditions as seen in Table 2. Note that this rate is out of the suggested optimal range 20-50% whereas that of model 1 was very close (19.97%) to the range. The increased model complexity considerably dropped convergence in terms of the Metropolis acceptance ratio. Second, as seen in Figure 4, the approximate burn-in size for this case is almost 3,000 which is 30 times more than that of the model 1. Third, Model 2 (e.g., 25.1 seconds) requires more than double CPU time than Model 1 (e.g., 12.3 seconds).

Also note that this model complexity issue become more critical with bad (i.e., non-optional or un-tuned) starting values and/or jump sizes as seen in Table 2. For example, for both arbitrary starting value with medium jump sizes and ML starting values with large jump sizes cases, the Metropolis acceptance rates were 0%. In other words, the Metropolis sampler does not accept any proposed values during the entire MCMC run with 10,000 iterations. Thus, the burn-in period should be larger than 10,000 iterations. These results show how the model complexity can critically impact the MCMC estimation and underscore that it is imperative to develop "tuned" MCMC algorithms for SEM models, especially given the complexity of many SEM models in practice.

Table 3 Arbitrary and ML starting values for both Model 1 and Model 2

	Model 1		**Model 2**	
Parameter	**Arbitrary value**	**ML value**	**Arbitrary value**	**ML value**
b_{21}	0	0.763	0	0.766
b_{31}	0	0.706	0	0.716
b_{52}	---	---	0	0.918
b_{62}	---	---	0	0.904
Var(F_1)	1	6.143	1	6.079
a_{21}	---	---	1	-0.365
Var(D)	---	---	1	3.506
Var(E_1)	1	0.839	1	0.903
Var(E_2)	1	2.472	1	2.478
Var(E_3)	1	1.978	1	1.922
Var(E_4)	---	---	1	1.252
Var(E_5)	---	---	1	1.693
Var(E_6)	---	---	1	1.450

Table 4 Iteration history for Model 1 (One factor Model) under medium jump size and ML starting values

Iteration #	**Parameter estimates**						**Likelihood**	**MCMC parameters**		
	b_{21}	b_{31}	**Var(E_1)**	**Var(E_2)**	**Var(E_3)**	**Var(E_4)**	$W(\mathbf{S}, \Sigma(\theta^c))$	α	u	**Decision**
0	0.76	0.71	0.84	2.47	1.98	6.14	1.42			
1	0.77	0.70	0.67	1.97	1.93	6.66	0.17	0.12	0.12	Accept
2	0.66	0.68	0.30	1.94	1.59	7.20	0.00	0.00	0.66	Reject
3	0.75	0.69	0.68	1.99	1.98	6.33	0.25	1.50	0.34	Accept
4	0.68	0.60	0.57	2.32	1.26	5.64	0.00	0.00	0.93	Reject
5	0.70	0.74	0.59	1.54	2.09	7.35	0.00	0.00	0.16	Reject
6	0.77	0.61	0.82	2.37	2.48	6.13	0.14	0.57	0.27	Accept
7	0.88	0.71	0.87	2.71	2.17	5.69	0.14	0.98	0.66	Accept
8	0.66	0.73	1.21	2.50	2.42	4.95	0.01	0.04	0.40	Reject
9	0.95	0.78	0.41	2.56	2.10	6.21	0.00	0.00	0.14	Reject
10	0.90	0.66	0.15	2.48	2.32	4.97	0.00	0.00	0.34	Reject
11	0.80	0.77	1.53	2.74	2.62	4.25	0.00	0.01	0.30	Reject
12	0.90	0.63	1.28	3.33	2.58	5.94	0.00	0.00	0.44	Reject

Table continued...

Iteration #	Parameter estimates						Likelihood	MCMC parameters		
	b_{21}	b_{31}	Var(E_1)	Var(E_2)	Var(E_3)	Var(E_4)	$W(S,\Sigma(\theta^{c}))$	α	u	**Decision**
13	0.94	0.65	0.98	2.68	2.14	6.07	0.01	0.07	0.11	Reject
14	0.80	0.80	0.53	2.45	1.81	5.03	0.01	0.08	0.98	Reject
15	0.84	0.75	0.79	2.97	2.55	4.88	0.01	0.09	0.97	Reject
16	0.88	0.65	1.10	2.61	2.19	4.53	0.01	0.09	0.21	Reject
17	0.86	0.63	1.44	2.74	2.55	5.01	0.00	0.02	0.68	Reject
18	0.80	0.69	0.95	2.27	2.06	6.28	0.91	6.42	0.04	Accept
19	0.86	0.63	0.69	2.62	1.91	5.94	0.06	0.06	0.71	Reject
20	0.85	0.82	0.14	2.49	2.41	6.63	0.00	0.00	0.36	Reject
Step 1 / 6	Step 2							Step 3	Step 4	Step 5

Note At iteration 0, the metropolis sampler is initialized, which is step 1 of each iteration in the metropolis algorithm. The last row indicates which steps in each iteration the numbers correspond to

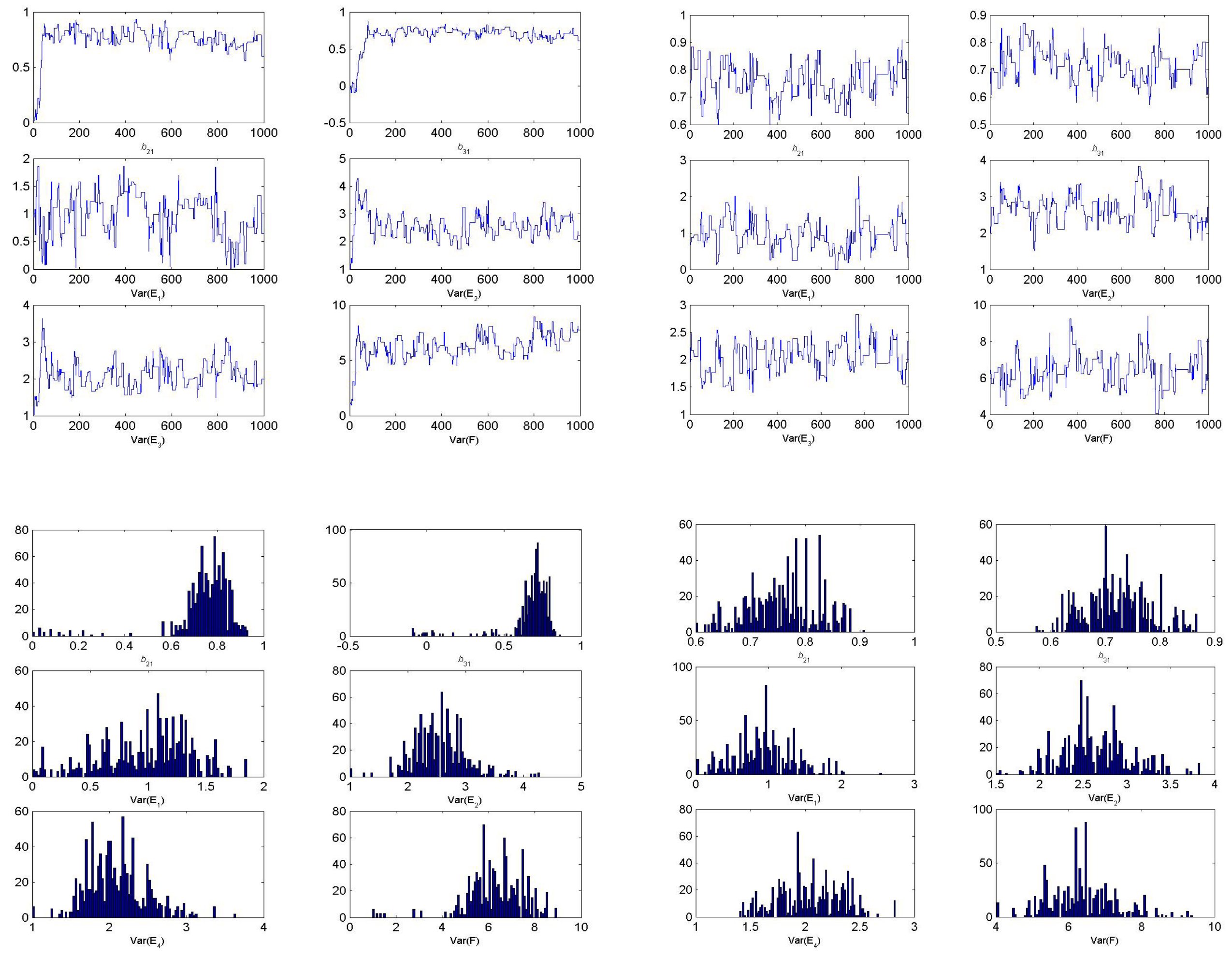

Figure 2 MCMC Chain histories and posterior densities for Model 1 (One factor model) parameters under arbitrary and ML starting values.

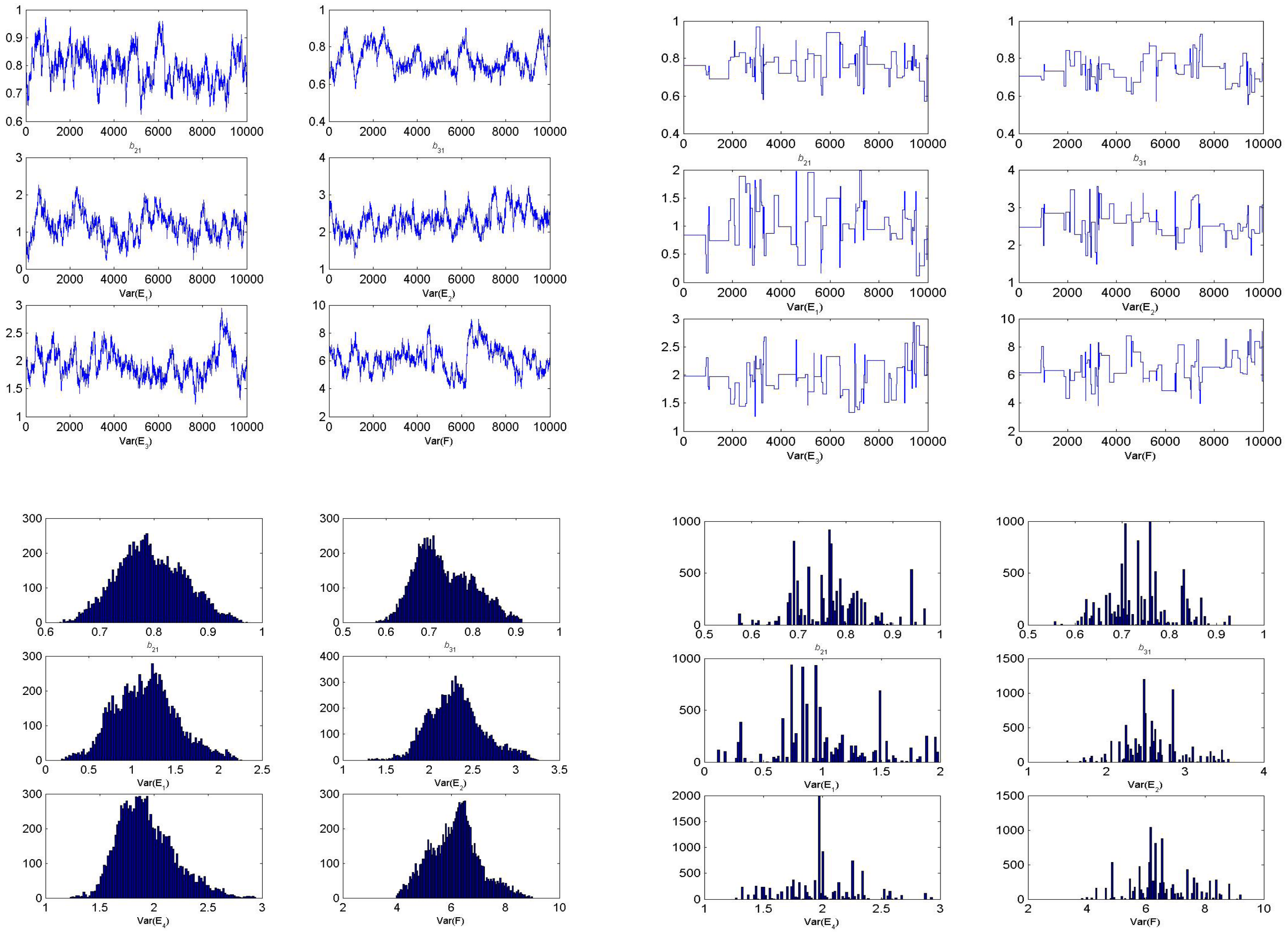

Figure 3 MCMC chain histories and posterior densities for Model 1 (One factor model) parameters under small and large jump sizes.

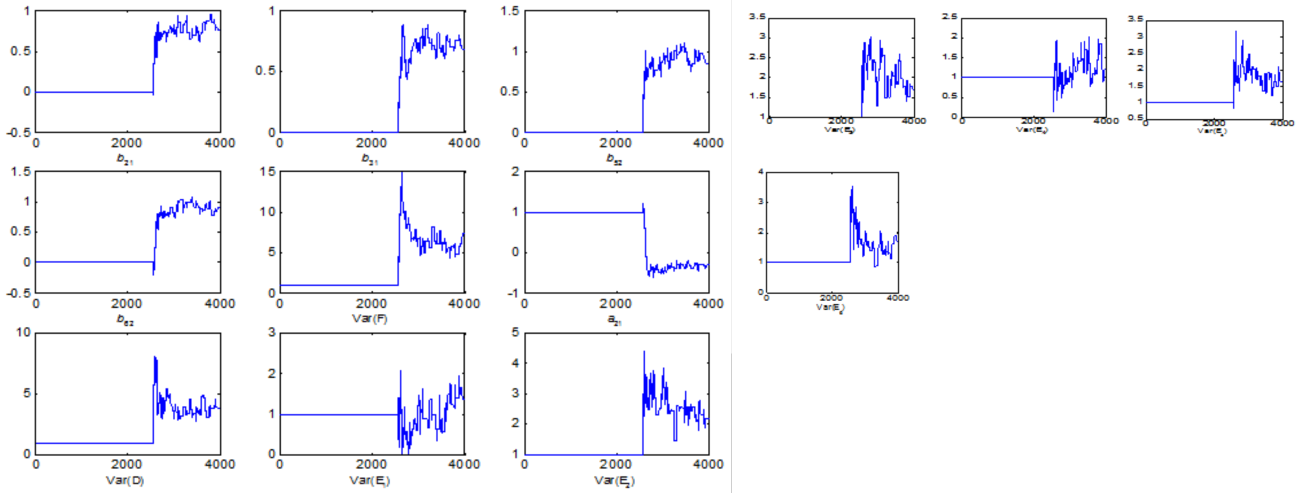

Figure 4 The first MCMC 4,000 iterations for Model 2 (Two factor Model) parameters under medium jump sizes and arbitrary starting values.

Table 5 ML and MCMC estimation summary results for Model 1 (One factor Model)

	Estimation details	**Statistic**	**Model parameter estimate**					
			b_{21}	b_{31}	**Var(E_1)**	**Var(E_2)**	**Var(E_3)**	**Var(F)**
ML		ML Esti.	0.763	0706	0.839	2.472	1.978	6.143
	ML iterations: 4	ML *SE*	0.076	0.069	0.429	0.385	0.316	.935
	Time: 0.0 Sec.	95% CI_L	0.614	0.571	-0.002	1.717	1.359	4.310
		95% CI_U	0.912	0.841	1.680	3.227	2.597	7.976
MCMC	Starting value: ML	*M*	0.769	0.714	0.943	2.590	2.027	6.259
		ME	0.770	0.714	0.943	2.575	2.016	6.182
	Jump size: Medium	*MIN*	0.545	0.486	0.006	1.267	1.132	3.808
	Iterations: 10,000	*MAX*	1.115	0.925	2.549	4.559	3.977	10.332
	Acceptance ratio: 19.97%	*SD*	0.077	0.066	0.396	0.411	0.323	0.923
		$PS_{2.5\%}$	0.624	0.590	0.195	1.812	1.430	4.524
	Time: 12.3 Sec.	$PS_{97.5\%}$	0.923	0.847	1.782	3.459	2.723	8.251
	Starting value: ML	*M*	0.770	0.740	1.016	2.572	1.968	6.504
		ME	0.770	0.734	0.944	2.504	1.978	6.312
	Jump size: Large	*MIN*	0.573	0.555	0.111	1.492	1.263	3.804
	Iterations: 10,000	*MAX*	0.969	0.930	1.987	3.561	2.934	9.230
	Acceptance ratio: 0.84%	*SD*	0.076	0.068	0.418	0.351	0.315	1.013
		$PS_{2.5\%}$	0.639	0.620	0.261	1.813	1.330	4.663
	Time:12.8 Sec.	$PS_{97.5\%}$	0.939	0.867	1.955	3.428	2.612	8.801
	Starting value: ML	*M*	0.794	0.734	1.159	2.308	1.935	6.142
		ME	0.790	0.722	1.159	2.296	1.899	6.184
	Jump size: Small	*MIN*	0.626	0.576	0.182	1.289	1.227	3.965
	Iterations: 10,000	*MAX*	0.973	0.913	2.265	3.260	2.942	9.004
	Acceptance ratio: 89.24%	*SD*	0.062	0.066	0.359	0.312	0.263	0.912
		$PS_{2.5\%}$	0.680	0.627	0.512	1.774	1.517	4.456
	Time:12.1 Sec.	$PS_{97.5\%}$	0.917	0.870	1.938	3.015	2.542	8.103
	Starting value: Arbitrary	*M*	0.747	0.676	0.974	2.547	2.091	6.305
		ME	0.767	0.712	1.035	2.518	2.052	6.313
	Jump size: Medium	*MIN*	0.000	-0.095	0.006	1.000	1.000	0.972
	Iterations: 1,000	*MAX*	0.933	0.867	1.853	4.281	3.639	8.953
	Acceptance ratio: 21.5%	*SD*	0.136	0.153	0.384	0.454	0.375	1.211
		$PS_{2.5\%}$	0.195	0.037	0.098	1.721	1.515	3.113
	Time:1.25 Sec.	$PS_{97.5\%}$	0.890	0.802	1.599	3.483	2.949	8.505
	Starting value: Arbitrary	*M*	0.770	0.714	0.960	2.566	2.044	6.251
		ME	0.768	0.713	0.956	2.537	2.029	6.192
	Jump size: Large	*MIN*	-0.593	-0.015	0.001	1.000	0.623	0.619
	Iterations: 1,000,000	*MAX*	1.156	1.061	2.936	5.100	3.681	10.832
	Acceptance ratio: 0.81%	*SD*	0.079	0.070	0.433	0.402	0.325	0.957
		$PS_{2.5\%}$	0.625	0.582	0.136	1.851	1.466	4.581
	Time:22.4 Min.	$PS_{97.5\%}$	0.931	0.854	1.825	3.418	2.742	8.337
	Starting value: Arbitrary	*M*	0.766	0.711	0.915	2.579	2.058	6.269
		ME	0.764	0.708	0.900	2.556	2.041	6.209
	Jump size: Small	*MIN*	-0.014	-0.032	0.000	1.000	0.820	1.000
	Iterations: 1,000,000	*MAX*	1.093	1.134	2.726	4.821	3.773	11.462
	Acceptance ratio: 88.71%	*SD*	0.077	0.072	0.428	0.394	0.333	0.982
		$PS_{2.5\%}$	0.623	0.581	0.146	1.868	1.457	4.511
	Time:22.3 Min.	$PS_{97.5\%}$	0.923	0.860	1.804	3.409	2.761	8.366

Note M = Mean of posterior distribution, *ME* = Median of posterior distribution, $PS_{2.5\%}$ = 2.5th percentile of posterior distribution, $PS_{97.5\%}$ = 97.5th percentile of posterior distribution

Discussion and conclusion

This article provided an accessible illustration of how different estimation choices impact how efficiently the Metropolis algorithm for MCMC estimation works in practice. Starting values for model parameters, jump sizes for the proposal distribution in the sampler, and complexity of the SEM model were varied in this paper to understand the consequence of these estimation conditions for the MCMC estimation results. The results in this paper showed how un-tuned starting values (i.e., an arbitrary starting value) and un-tuned jump sizes (i.e., jump sizes that are either too small or too large) for the proposal distribution that are result in slow and/or poor convergence.

The results show that "tuning" a sampling algorithm to achieve good mixing for a complex model is a challenging task, both computationally and mathematically (Gilks et al.,[30]). Since the Metropolis algorithm has been introduced, more than dozens of MCMC algorithms have been developed during last three decades, which represent refinements of the algorithm for particular model structures. These algorithms include, among others, the Metropolis-Hastings algorithm (Hastings,[40]), the Gibbs sampler (Gelfand & Smith;[41] Geman, & Geman,[42]), and the hybrid Monte Carlo algorithm (Duane, et al.,[43]).

What this paper has shown is it is imperative to develop "tuned" MCMC algorithms so that they are well fitted for estimating SEM models, especially given the complexity of popular SEM models in practice. There currently is a practical disconnect between the technical advantages of MCMC over gradient-based estimation approaches, which have been well documented for decades, and meaningful advice for practitioners who want to understand how MCMC estimation works.

This paper was only illustrative, however. As such, it is limited in scope because (a) it discussed only the Metropolis algorithm, (b) used only the Wishart likelihood function, (c) used only a normal distribution function, (d) used only two simple SEM models, (e) used only selected estimation choices, and (f) focused only on selected outcome graphics and statistics. Nevertheless, as an illustrative paper, it makes two contributions to the emerging SEM literature on MCMC estimation. First, both numerical and conceptual details of MCMC estimation mechanisms with hands-on examples were provided to enhance the practical know-how of MCMC estimation for SEM specialists and to provide a more intuitive understanding of MCMC estimation process. Second, this article provided information regarding the consequence of various facets of MCMC estimation as a consciousness-raising device to illustrate that implementing MCMC estimation in latent variable modelling area is not as straightforward as some authors (e.g., Patz & Junker,[44-46]) make it seem.

Acknowledgements

The author would like to thank Dr. Sunhee Kim and Dr. Andre Rupp for the valuable comments on the earlier version of this manuscript.

Conflicts of interest

None.

References

1. Bollen KA. Structural equation modeling with latent variables. Wiley, USA. 1989.
2. Browne MW. Generalized least-squares estimators in the analysis of covariance structures. *South African Statistical Journal*. 1974;8:1–24.
3. Browne MW. Asymptotically distribution-free methods for the analysis of covariance structures. *British Journal of Mathematical and Statistical Psychology*. 1984;37(1):62–83.
4. Gelman A, Carlin JB, Stern HS, et al. Bayesian data analysis. Chapman/Hall, USA, 1995.
5. Lee SY. Structural Equation Modeling: A Bayesian Approach. West Sussex: John Wiley & Sons Ltd. 2007.
6. Choi J, Levy R. Markov chain Monte Carlo Estimation Methods for Structural Equation Modeling: A Comparison of Subject-level Data and Moment-level Data Approaches. *Biometrics and Biostatistics International Journal*. 2017;6(5):00182.
7. Levy R, Mislevy RJ. Bayesian Psychometric Modeling. FL: Chapman & Hall/CRC, Boca Raton, USA, 2016.
8. Junker B. Beyond MCMC. Paper presented at the annual meeting of the Psychometric Society (IMPS), USA, 2008.
9. Chen J, Choi J, Stapleton L, et al. An Empirical Evaluation of Mediation Effect Analysis with Manifest and Latent Variables using Markov chain Monte Carlo and Alternative Estimation Methods. Structural Equation Modeling: A Multidisciplinary Journal. 2014;21(2).
10. Kim JS, Bolt DM. Estimating item response theory models using Markov chain Monte Carlo methods. *Educational Measurement: Issues and Practice Volume*. 2007;26(4):38–51.
11. Lynch SM. Introduction to applied Bayesian statistics and estimation for social scientists. Springer, New York, USA, 2007.
12. Spiegelhalter DJ, Thomas A, Best NG, et al. BUGS: Bayesian inference Using Gibbs Sampling, Version 0.5, (version ii). Cambridge Medical Research Council Biostatics Unit. 1996.
13. Arbuckle JL. Amos 7.0 User's Guide. SPSS Inc, USA. 2006.
14. Metropolis N, S Ulam S. The Monte Carlo method. *Journal of American Statistical Association*. 1949;44(247):335–341.
15. Metropolis N, Rosenbluth AW, Rosenbluth MN, et al. Equations of state calculations by fast computing machines. *Journal of Chemical Physics*. 1953;21(6):1087–1091.
16. Jöreskog KG. A general approach to confirmatory maximum likelihood factor analysis. *Psychometrika*. 1969;34(2):183–202.
17. Kennedy WJ, Gentle JE. Statistical computing. Marcel Dekker, New York, USA. 1980.
18. Süli E, Mayers D. An Introduction to Numerical Analysis. Cambridge University Press, India. 2003.
19. Choi J, Kim S, Chen J, et al. A Comparison of Maximum Likelihood and Bayesian Estimation for Polychoric Correlation using Monte Carlo Simulation. *Journal of Educational and Behavioral Statistics*. 2011;36(4):523–549.
20. Choi J, Dunlop M, Chen J, et al. A Comparison of Different Approaches for Coefficient Alpha for Ordinal Data. *Journal of Educational Evaluation*. 2011;24(2):485–506.
21. Andrieu C, Freitas N De, Doucet NA, et al. An introduction to mcmc for machine learning. *Machine Learning*. 2003;50:5–43.
22. Dyer M, Frieze A, Kannan R. A random polynomial-time algorithm for approximating the volume of convex bodies. *Journal Association Computing Machines*. 1991;38(1):1–17.

23. Jerrum M, Sinclair A. The Markov chain Monte Carlo method: an approach to approximate counting and integration. In: Hochbaum DS (Ed.), Approximation algorithms for NP-hard problems, PWS Publishing. *Massachusetts*. 1996;482–519.
24. Arminger G, Muthén BO. A Bayesian approach to nonlinear latent variable models using the Gibbs sampler and the Metropolis-Hastings algorithm. *Psychometrika*. 1998;63(3):271–300.
25. Lee SY, Song XY. Model comparison of nonlinear structural equation models with fixed covariates. *Psychometrika*. 2003a;68(1):27–47.
26. Lee SY, Song XY. Bayesian model selection for mixtures of structural equation models with an unknown number of components. *Br J Math Stat Psychol*. 2003b;56(Pt 1):145–165.
27. Scheines R, Hoijtink H, Boomsma A. Bayesian estimation and testing of structural equation models. *Psychometrika*. 1999;64(1):37–52.
28. Shi JQ, Lee SY. Bayesian sampling-based approach for factor analysis models with continuous and polytomous data. *British Journal of Mathematical and Statistical Psychology*. 1998;51(2):233–252.
29. Song XY, Lee SY. Bayesian estimation and test for factor analysis model with continuous and polytomous data in several populations. *British Journal of Mathematical and Statistical Psychology*. 2001;54(2):237–263.
30. Song X, Lee SY. A Bayesian approach for multigroup nonlinear factor analysis. *Structural Equation Modeling: A Multidisciplinary Journal*. 2002;9(4):523–553.
31. Gilks WR, Richardson S, Spiegelhalter DJ. Introducing Markov chain Monte Carlo. In: Gilks WR & Richardson S (Eds.), Markov chain Monte Carlo in practice, Chapman and Hall. 1996;1–19.
32. Levy R, Choi J. An introduction to Bayesian Structural Equation Modeling. In: GR Hancock & RO Mueller (Eds,) Structural Equation Modeling: A Second Course (2nd edn), CT: Information Age Publishing, Inc, Greenwood, India. 2013.
33. Tierney L. Markov chains for exploring posterior distributions. *The Annals of Statistics*. 1994;22(4):1701–1762.
34. MATLAB 6.5. MathWorks Inc, Natick, Massachussetts, USA. 2002.
35. Bentler PM, Wu EJC. EQS 6.1 for Windows. Encino, CA, USA. 2004.
36. Reisenzein R. A structural equation analysis of Weiner's attribution-affect model of helping behavior. *Journal of Personality and Social Psychology*. 1986;50:1123–1133.
37. Weiner B. May I borrow your class notes? An attributional analysis of judgments of help giving in an achievement-related context. *Journal of Educational Psychology*. 1980;72(5):676–681.
38. Roberts GO, Gelman A, Gilks WR. Weak Convergence and Optimal Scaling of Random Walk Metropolis Algorithms. *Annals of Applied Probability*. 1997;7(1):110–120.
39. Roberts GO, Rosenthal JS. Optimal scaling of discrete approximations to Langevin diffusions. *Journal of the Royal Statistical Society*, Series B, Methodological. 1998;60(1):255–268.
40. Hastings WK. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*. 1970;57(1):97–109.
41. Gelfand AE, Smith AFM. Sampling-based approaches to calculating marginal densities. *Journal of the American Statistical Association*. 1990;85:398–409.
42. Geman S, Geman D. Stochastic relaxation, Gibbs distribution and Bayesian restoration of images. *IEE Transactions on Pattern Analysis and MachineIntelligence*. 1984;6(6):721–741.
43. Duane S, Kennedy AD, Pendleton BJ, et al. Hybrid Monte Carlo. Physics Letters B. 1987;195(2):216–222.
44. Patz RJ, Junker BW. Applications and extensions of MCMC in IRT: Multiple item types, missing data, and rated responses. *Journal of Educational and Behavioral Statistics*. 1999a;24(24):342–366.
45. Patz RJ, Junker BW. A straight forward approach to Markov Chain Monte Carlo methods for item response models. *Journal of Educational and Behavioral Statistics*. 1999b;24(2):146–178.
46. Lee SY, Song XY. Evaluation of the Bayesian and maximum likelihood approaches in analyzing structural equation models with small sample sizes. *Multivariate Behavioral Research*. 2004;39(4):653–686.