Fitting a Structural Equation Model

James H. Steiger

Department of Psychology and Human Development
Vanderbilt University

Fitting a Structural Equation Model

Introduction

- Suppose we have a structural equation model, and some data that we think the model might be appropriate for.
- What steps are involved in:
  1. Communicating the model to the computer?
  2. Obtaining the "best fitting" solution for the model parameters by iteration?
  3. Evaluating model fit?
- In many cases, one might be able, using modern software, to perform these steps without much (if any) understanding of how they are in fact accomplished.
- However, your decisions about what "buttons to press" in your structural equation modeling software may be more informed, and may lead to better answers, if you have some basic understanding of how these steps are performed.
- In this module, we'll look at a simple confirmatory factor model and discuss the steps involved in fitting it.

A Simple Covariance Structure Model

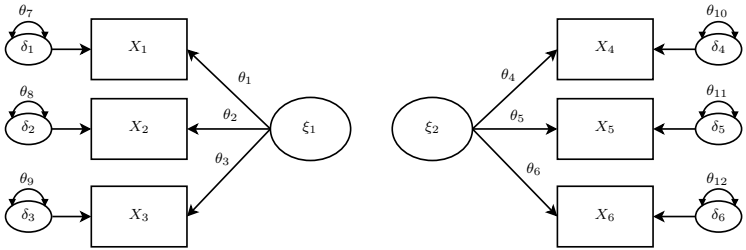
- Consider the simple confirmatory factor model shown on the next slide.
- It shows two orthogonal factors, each loading on 3 manifest variables.
- Since there are no slings for the factors, they have an assumed variance of 1.
- We already know that this factor model can be written

$$\mathbf{x} = \boldsymbol{\Lambda}\boldsymbol{\xi} + \boldsymbol{\delta} \tag{1}$$

and that, at the variance-covariance level, the model may be written in the classic LISREL notation as

$$\boldsymbol{\Sigma}_{xx} = \boldsymbol{\Lambda}\boldsymbol{\Lambda}' + \boldsymbol{\Theta}_{\delta} \tag{2}$$

A Simple Covariance Structure Model



A Simple Covariance Structure Model

- The model is a covariance structure model. The classic null hypothesis in covariance structure modeling is written in general form as

$$\boldsymbol{\Sigma} = \mathbf{M}(\boldsymbol{\theta}) \tag{3}$$

  where $\boldsymbol{\theta}$ is a vector of free parameters to be estimated, and $\mathbf{M}()$ is a model matrix function that carries the elements of $\boldsymbol{\theta}$ into the symmetric matrix $\boldsymbol{\Sigma}$.
- In the case of our confirmatory factor model, we may easily insert the 12 elements of $\boldsymbol{\theta}$ into $\boldsymbol{\Lambda}$ and $\boldsymbol{\Theta}_{\boldsymbol{\delta}}$ and compute the elements of $\mathbf{M}(\boldsymbol{\theta})$.

A Simple Covariance Structure Model

$$\boldsymbol{\Lambda}(\boldsymbol{\theta}) = \begin{bmatrix} \theta_1 & 0 \\ \theta_2 & 0 \\ \theta_3 & 0 \\ 0 & \theta_4 \\ 0 & \theta_5 \\ 0 & \theta_6 \end{bmatrix}, \quad \boldsymbol{\Theta}_{\boldsymbol{\delta}}(\boldsymbol{\theta}) = \begin{bmatrix} \theta_7 & 0 & 0 & 0 & 0 & 0 \\ 0 & \theta_8 & 0 & 0 & 0 & 0 \\ 0 & 0 & \theta_9 & 0 & 0 & 0 \\ 0 & 0 & 0 & \theta_{10} & 0 & 0 \\ 0 & 0 & 0 & 0 & \theta_{11} & 0 \\ 0 & 0 & 0 & 0 & 0 & \theta_{12} \end{bmatrix} \tag{4}$$

A Simple Covariance Structure Model

$$\mathbf{M}(\boldsymbol{\theta}) = \boldsymbol{\Lambda}(\boldsymbol{\theta})\boldsymbol{\Lambda}'(\boldsymbol{\theta}) + \boldsymbol{\Theta}_{\boldsymbol{\delta}}(\boldsymbol{\theta}) \tag{5}$$

$$= \begin{bmatrix} \theta_1^2+\theta_7 & \theta_2\theta_1 & \theta_3\theta_1 & 0 & 0 & 0 \\ \theta_2\theta_1 & \theta_2^2+\theta_8 & \theta_3\theta_2 & 0 & 0 & 0 \\ \theta_3\theta_1 & \theta_3\theta_2 & \theta_3^2+\theta_9 & 0 & 0 & 0 \\ 0 & 0 & 0 & \theta_4^2+\theta_{10} & \theta_5\theta_4 & \theta_6\theta_4 \\ 0 & 0 & 0 & \theta_5\theta_4 & \theta_5^2+\theta_{11} & \theta_6\theta_5 \\ 0 & 0 & 0 & \theta_6\theta_4 & \theta_6\theta_5 & \theta_6^2+\theta_{12} \end{bmatrix} \tag{6}$$

Evaluating Model Fit with a Discrepancy Function

- Given a model (and its matrix function), any set of parameter estimates may be evaluated by
  1. Substituting the elements of $\boldsymbol{\theta}$ into the model matrices and computing the model matrix function, then
  2. Comparing the model-reproduced covariance matrix with the sample covariance matrix.
- This comparison is accomplished through a discrepancy function.

Discrepancy Functions

- In what follows, the model function $\mathbf{M}(\boldsymbol{\theta})$ is assumed to be a twice differentiable function of the t free parameters in the vector $\boldsymbol{\theta}$.
- The discrepancy function $F(\mathbf{S}, \mathbf{M}(\boldsymbol{\theta}))$ is a measure on $\mathbf{S}$ and $\mathbf{M}(\boldsymbol{\theta}))$. The following 3 restrictions will lead to consistent estimates for the elements of $\boldsymbol{\theta}$
  1. $F(\mathbf{S}, \mathbf{M}(\boldsymbol{\theta})) \geq 0$
  2. $F(\mathbf{S}, \mathbf{M}(\boldsymbol{\theta})) = 0$ if and only if $\mathbf{S} = \mathbf{M}(\boldsymbol{\theta})$
  3. $F(\mathbf{S}, \mathbf{M}(\boldsymbol{\theta}))$ is continuous in $\mathbf{S}$ and $\mathbf{M}(\boldsymbol{\theta})$

Discrepancy Functions

The Ordinary Least Squares (OLS) Discrepancy Function

- This Ordinary Least Squares (OLS) discrepancy function is simply half the sum of squared differences between the elements of the sample covariance matrix and the model-reproduced covariance matrix, i.e.,

$$F_{OLS}(\mathbf{S}, \mathbf{M}(\boldsymbol{\theta})) = \frac{1}{2}\,\mathrm{Tr}(\mathbf{S} - \mathbf{M}(\boldsymbol{\theta}))^2 \tag{7}$$

- The values of $\boldsymbol{\theta}$ that minimize the OLS discrepancy function are called OLS estimates.
- OLS estimates have some problems:
  1. The OLS discrepancy is not scale free - different scalings of the manifest variables can produce different discrepancy function values.
  2. Moreover, when calculated on sample discrepancies, simple sums of squares may be inappropriate from a statistical standpoint, because the elements of S are not independent random variables, and because they usually have different sampling variances.

Discrepancy Functions

The Generalized Least Squares (GLS) Discrepancy Function

- The Generalized Least Squares (GLS) Discrepancy Function compensates for the problems of OLS estimates by, in effect, standardizing the sample discrepancies.
- This function is

$$F_{GLS}(\mathbf{S}, \mathbf{M}(\boldsymbol{\theta})) = \frac{1}{2} \mathrm{Tr}((\mathbf{S} - \mathbf{M}(\boldsymbol{\theta}))\mathbf{S}^{-1})^2 \tag{8}$$

- GLS estimates are values in $\boldsymbol{\theta}$ that minimize the GLS discrepancy function.

Discrepancy Functions

The Iteratively Reweighted Generalized Least Squares (IRGLS) Discrepancy Function

- The Iteratively Reweighted Generalized Least Squares (IRGLS) Discrepancy Function updates the weights applied to the discrepancies on each iteration.
- This function is

$$F_{IRGLS}(\mathbf{S}, \mathbf{M}(\boldsymbol{\theta})) = \frac{1}{2}\,\mathrm{Tr}((\mathbf{S} - \mathbf{M}(\boldsymbol{\theta}))\mathbf{M}(\boldsymbol{\theta})^{-1})^2 \tag{9}$$

- This discrepancy function is used in a Gauss-Newton algorithm as an efficient way to obtain Maximum Wishart Likelihood estimates.

Discrepancy Functions

The Maximum Wishart Likelihood (ML) Discrepancy Function

- A more complex function is the Maximum Wishart Likelihood (ML) discrepancy function. This function may be written

$$F_{ML}(\mathbf{S}, \mathbf{M}(\theta)) = \log|\mathbf{M}(\theta)| - \log|\mathbf{S}| + \mathrm{Tr}(\mathbf{S}\mathbf{M}(\theta)^{-1}) - p \tag{10}$$

  witn p is the number of manifest variables.
- If $\mathbf{S}$ has a Wishart distribution (a somewhat less restrictive assumption than the requirement that the observed variables follow a multivariate normal distribution), minimizing the ML discrepancy function produces Maximum Wishart Likelihood Estimates, generally referred to as Maximum Likelihood estimates in the structural equation modeling literature.

The Chi-Square Test of Fit

- If $\mathbf{S}$ has a Wishart distribution, the model is identified, and $\boldsymbol{\theta}$ has t free parameters, then under fairly general conditions $(n-1)F_{ML}(\mathbf{S}, \boldsymbol{\sigma}(\boldsymbol{\theta}))$, and $(n-1)F_GLS(\mathbf{S}, \boldsymbol{\sigma}(\boldsymbol{\theta}))$ both have an asymptotic χ^2 distribution with $p(p+1)/2 - t$ degrees of freedom.
- Such a χ^2 statistic, often described as a "goodness-of-fit" statistic (but perhaps more accurately called a "badness-of-fit" statistic) allows us to test statistically whether a particular model fits $\boldsymbol{\Sigma}$ perfectly in the population, i.e., whether $\boldsymbol{\Sigma} = \mathbf{M}(\boldsymbol{\theta})$.
- There is a long tradition of performing such a test, although it is becoming increasingly clear that the procedure is seldom appropriate.

A Caution about Parameter Count

- As mentioned above, the degrees of freedom for the asymptotic χ^2 test statistic are $p(p+1)/2 - t$, where t is the number of free parameters in $\boldsymbol{\theta}$.
- Beginners are sometimes surprised when the degrees of freedom reported by their structural equation modeling program do not "add up."
- This can happen when the parameters in $\boldsymbol{\theta}$ are not *mathematically independent and variable*, i.e., some parameters are determinate functions of others and are not free to vary independently.

A Caution about Parameter Count

- An example of this is in unrestricted exploratory factor analysis, in which the elements of the $p \times m$ factor pattern $\mathbf{\Lambda}$ are not free to vary independently.
- In the handout *Confirmatory Factor Analysis with R* I demonstrate why a certain number of the elements of $\mathbf{\Lambda}$ may always be set equal to zero, and hence are not free to vary.
- For example, if one has p variables and m factors, there are $p \times m$ loadings p unique variances, and $m(m-1)/2$ factor intercorrelations to estimate. So with 9 variables and 3 factors, one might expect the parameter count to be $27 + 3 + 9 = 39$, and the degrees of freedom to be $p(p+1)/2 - 39 = 45 - 39 = 6$.
- However, in EFA, the degrees of freedom in general are actually

$$\frac{1}{2}((p-m)^2 - p - m) = 12$$

. So the number of "really free" parameters is 6 fewer than it seems.
- Fortunately, modern software can deduce the correct number of degrees of freedom during iteration in most cases.

Nonlinear Optimization

- We have our model function, and we have chosen our discrepancy function (let's say, ML).
- We are ready to find the best fitting $\boldsymbol{\theta}$.
- How is this done? By "iteration," or more formally, by nonlinear optimization with an iterative algorithm.

Nonlinear Optimization

- It turns out that some basic understanding of the nature of the nonlinear optimization process may give you a way out of the dreaded "Model failed to converge" error message that always seems to occur when you least expect it.
- It is a fact that structural equation modeling software, be it commercial or open-source or somewhere in-between, *always converges on all the examples in the manual* unless the manual deliberately contains an example of non-convergence in order to school the user on approriate "rescue measures."

Nonlinear Optimization

- As an example of the kind of approach typically taken, we'll examine the well-known Gauss-Newton approach, used in several commercial software programs.
- The iteration starts with initial estimates (often referred to as "starting values") for the elements of $\boldsymbol{\theta}$.
- On each iteration, the current value of the function is calculated, and the program estimates, using derivatives, which direction of change for $\boldsymbol{\theta}$ will produce a further decrease in the discrepancy function.

Nonlinear Optimization

- $\boldsymbol{\theta}$ is changed in that direction by an initial amount (called the "step length"), and the function is recalculated with this new $\boldsymbol{\theta}$.
- It may be that, according to certain criteria, the initial step went either too far, or not far enough. In that case the step length may go through several adjustments during an iteration.
- Once the "best" step length is estimated, the program moves on to the next iteration. When the discrepancy function stops improving, the algorithm terminates.
- Now for some technical details.

Nonlinear Optimization

- Let $\mathbf{d}(\boldsymbol{\theta})$ be a vector of first partial derivatives (of the elements of $\mathbf{M}(\boldsymbol{\theta})$ with respect to the elements of $\boldsymbol{\theta}$. That is,

$$\mathbf{d}(\boldsymbol{\theta}) = \partial Vec\mathbf{M}(\boldsymbol{\theta})/\partial\boldsymbol{\theta} \tag{11}$$

- In the Gauss-Newton approach $\mathbf{H}_k$ is an approximate Hessian given by

$$\mathbf{H}_k = 2\mathbf{d}'(\boldsymbol{\theta}_k)\mathbf{d}(\boldsymbol{\theta}) \tag{12}$$

  and $\mathbf{g}_k$ is the negative gradient of the discrepancy function with respect to $\boldsymbol{\theta}$, i.e,

$$\mathbf{g}_k = -\partial F(\mathbf{S}, \mathbf{M}(\boldsymbol{\theta}))/\partial\boldsymbol{\theta}\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k} \tag{13}$$

- Lee and Jennrich(1979, *Psychometrika*) showed that the Gauss-Newton step using the IRGLS estimator is equivalent to the Fisher scoring step

Nonlinear Optimization

- In the Gauss-Newton algorithm, the estimate on the kth iteration is related to the estimate on the next iteration by the formula

$$\begin{aligned} \boldsymbol{\theta}_{k+1} &= \boldsymbol{\theta}_k + \lambda_k \mathbf{H}_k^{-1} \mathbf{g}_\mathbf{k} && (14) \\ &= \boldsymbol{\theta}_k + \lambda_k \boldsymbol{\delta}_k && (15) \end{aligned}$$

  where $\boldsymbol{\delta}_k$ establishes the direction of change for the parameters on the kth iteration, while the scalar parameter λ_k establishes, jointly with $\boldsymbol{\delta}_k$, the length of the step vector.
- Especially during the early phases of iteration, the program may attempt to take extremely large steps.
- Often, this causes no problem, and in fact hastens the progress toward a correct solution.

Nonlinear Optimization

- However, sometimes large steps result in a set of parameter estimates that cause iteration to "blow up," either because
  1. The estimates yield a singular estimated covariance matrix, or
  2. Because the estimates end up in a region from which step direction is incorrect, and recovery is impossible.
- If the step direction is correct, but the step length is too large (or too small), the program can recover by changing the step length via an adjustment of λ.
- Notice that, once the step direction has been established by $\boldsymbol{\delta}$, the multidimensional optimization problem has been reduced to a unidimensional optimization problem in which the parameter search occurs only along the line established by $\boldsymbol{\delta}$, hence the term "line search."
- Optimizing a function in only one dimension is non-trivial, but a lot easier than multidimensional optimization.

Nonlinear Optimization

Line Search

- There are a number of classic algorithms available for the line search, including bisection, golden search, and cubic interpolation.
- Several software programs have used simple stephalving as the line search method.
- In this approach, a full step is taken initially (i.e., $\lambda = 1$). If this full step yields a reduction in the discrepancy function, the iteration cycle continues. If a reduction does not occur, λ is divided by 2, thus halving the step length, and $\boldsymbol{\theta}_k$ and the discrepancy function are recalculated.
- If this fails to produce an improvement, the step is halved again, etc.
- Although it often works very well, simple stephalving can fail, for an obvious reason (C.P.).

Nonlinear Optimization

Steepest Descent Iterations

- Sometimes, when full Gauss-Newton iteration fails, one can rescue the optimization by inserting some "steepest descent interations" at the beginning of the optimization.
- These are defined as

$$\boldsymbol{\theta}_{k+1} = \lambda_k \mathbf{g}_k + \boldsymbol{\theta}_k \tag{16}$$

Nonlinear Optimization

Jennrich-Sampson Modification

- Sometimes, during iteration, the program will step to a boundary of the parameter space.
- Other times, the approximate Hessian will become singular.
- Jennrich and Sampson (1968, *Psychometrika*) developed an ingenious procedure that can often solve both problems and keep the iteration going.
- Although it was originally developed in the context of non-linear regression, the method transfers beautifully to covariance structure modeling, and several programs use it.

Rescuing an Optimization – General Principles and Methods

- If iteration fails, there are a number of things you can try.
  1. If the iteration was still proceeding at the default maximum number of iterations, increase the maximum. Most if not all programs will allow you to do this.
  2. Examine your manifest variables. If there are huge differences in their variances, reduce them by multiplying or dividing by a simple constant like 5 or 10. *Do not standardize by dividing by the observed standard deviation.*
  3. Increase (or decrease) the number of steepest descent iterations.
  4. If your program offers the option, reduce the initial step length by starting λ at a value much less than 1. In my experience, the majority of failed iterations can be rescued with this simple method.
  5. Try to get different starting values. There are several ways you can do this.
     1. Try a different method of calculating starting values. Several programs have "default" values and "automatic" values. The latter may help in cases where the former fail.
     2. One is to try a different discrepancy function (OLS or GLS, for example) and, if that converges, use the estimates as starting values. Some programs offer the option to do this automatically.
     3. Another is to break your model into submodels and estimate each separately. In general, smaller models are easier to optimize than larger models.
  6. Use another program. One program may succeed where others fail.