

An Algorithm for the Hierarchical Organization of Path Diagrams and Calculation of Components of Expected Covariance

Steven M. Boker

Department of Psychology
The University of Notre Dame
Notre Dame, Indiana 46556

J. J. McArdle

Department of Psychology
The University of Virginia
Charlottesville, Virginia 22903

Michael Neale

Department of Psychiatry
Medical College of Virginia
Richmond, Virginia 22903

In press in *Structural Equation Modeling*

Abstract

Modern structural equation modeling software programs include user interfaces for the entry of graphical diagrams as a method for the production of the underlying matrices that are then manipulated in traditional ways to provide parameter estimates and fit statistics. This work presents an algorithm for the solution of the problem in the reverse order: automatically producing a graphical diagram from a matrix formula in such a way that its components are logically and hierarchically arranged. In the process, the individual path components of expected covariance between variables are calculated and feedback loops or so-called nonrecursive paths are recognized and tagged. Possible uses for this algorithm include automatic recognition of regions of structural underidentification in a model and the didactic graphical display of the components of expected covariance between variables. This algorithm relies on the matrix equations of McArdle and McDonald (1984), a general purpose SEM formulation.

Introduction

A number of structural equation modeling programs now include a graphical user interface that allows the creation of path diagrams and a means of building model matrices (Arbuckle, 1997; Bentler, 1995; Jöreskog & Sörbom, 1996; Muthén, 1998; Neale, Boker, Xie, & Maes, 1999). Although the authors are aware of no formal study, students report that these graphical user interfaces have reduced the time that it takes for a novice to learn how to specify a model of interest. However, many experts continue to specify models using model matrices for a variety of reasons including familiarity, control, and the ability to make minor modifications to existing scripts. An algorithm that automatically translated existing models specified through matrices into an organized and logical path diagram would be useful in presenting these model specifications for presentation or publication.

The graphical method of building model matrices requires there be no ambiguity in model specification when the path diagram is drawn. That is to say, one diagram must result in an unambiguous set of model matrices. Of course, there are equivalent matrix formulations that can be used for specifying the same model specification. As long as the path diagram to model matrix translation can produce one of these equivalent formulations, an appropriate estimation program can fit the model to the data.

Similarly, an automatic translation of a model specification from its matrix form into a path diagram relies on diagrammatic conventions such that there is no ambiguity in the resulting path diagram. Thus one model specification must be able to be mapped onto one structurally unambiguous diagram. Thus the variables and the paths connecting the variables can have only one configuration topologically (Gemignani, 1972). However, one may place the variables anywhere on the graphical page. Almost all random choices of placement of variables on the page will produce a visually confusing diagram once the paths are drawn (see Figure 1).

When a path diagram is drawn well, there is an order to the placement of variables that facilitates interpretation of the model (see Figure 2). One simple heuristic posits that variables that are directly connected to each other should be placed near each other. Another heuristic that is commonly used is that of a hierarchy of variables, for instance manifest variables (observed variables), first order factors and second order factors. We will take advantage of these two heuristics to construct an algorithm that can be used to supply information about where variables should be logically placed on a page.

Graph theory has developed a number of concepts that are useful in thinking abstractly about a path diagram. In graph theory, a path diagram would be called a *digraph*, since it is composed of *nodes* (circles and boxes) and *edges* (one and two headed arrows), and since at least some of the edges have a direction (regression weights) (see Christofides, 1975, for an introduction). The *span* of a graph denotes the number of nodes crossed by the longest directed path in a digraph. If each edge is of a length greater than or equal to one, the graph will cover a minimal area when the nodes are placed on a page such that the length of the edges are minimized. In this case, the length of the span of the graph will be similar to the greatest geometric distance between any two variables.

One way to organize a path diagram so as to (a) place variables that are closely connected near each other, and (b) create a hierarchy of variables; is to calculate the number of nodes crossed by the longest path between every pair of variables and then arrange the

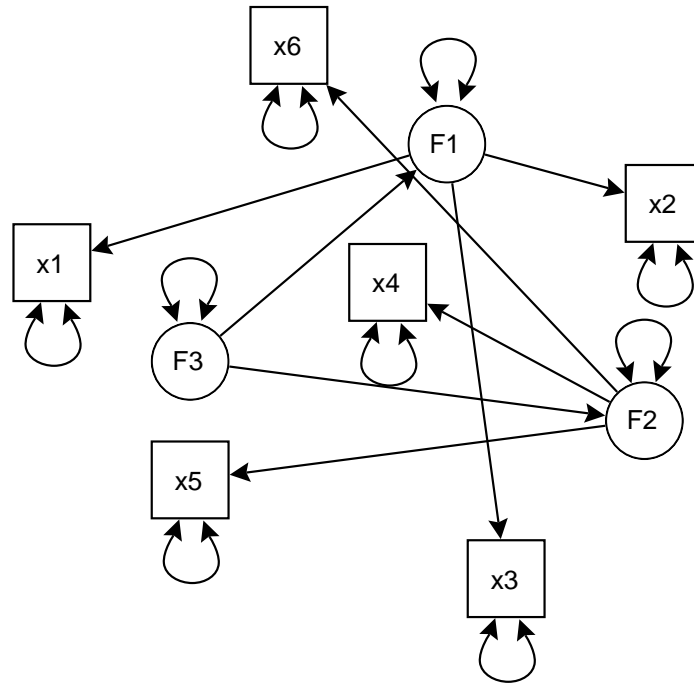


Figure 1. A confusing path diagram resulting from random placement of variables on the page.

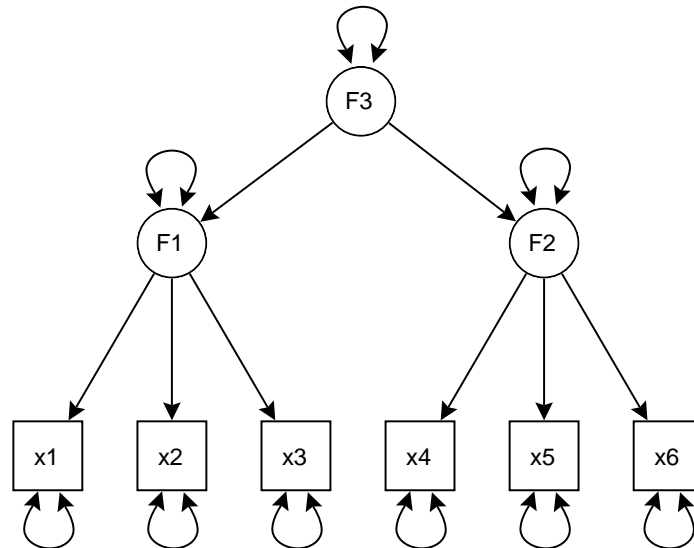


Figure 2. The same model as Figure 1 but organized more effectively.

variables on the page in such a way that the geometric distance between variables is as similar as possible to the length of the longest path. This solution effectively keeps variables that are directly connected as close as possible to each other while at the same time can be used to create a hierarchy of variables. One could visualize this solution as picking up the diagram in Figure 1 by the variable $F3$, giving it a quick shake, and then laying it carefully back down onto the page.

The hierarchy of variables generated by the proposed algorithm consists at the lowest level of variables (Level 0) that have no directed arrows leaving them. At Level 1, there are variables that have at least one arrow leaving them that terminates in a variable at Level 0, but no arrow that terminates at higher levels. At Level 2, there are variables that have at least one arrow leaving them that terminates in a variable at Level 1, no arrows terminating at higher levels, and possibly some arrows that terminate at Level 0. In this way, the length of the longest (possibly mediated) path emanating from a variable determines the level at which it belongs.

In order for this algorithm to be complete, it must take into account non-recursivity, directed paths that describe feedback loops. First, feedback loops of directed paths must be recognized. We have taken the approach that all variables in a feedback loop should be kept on the same level where the level to which the feedback loop is assigned is determined by the longest path emanating from any variable in the loop.

In order implement this solution, one must calculate the longest path connecting all pairs of variables in a model. The proposed solution to this problem has two additional benefits: (1) it can be used to isolate and calculate all of the components of the covariance between any two variables, and (2) it recognizes and tags feedback loops. The remainder of the article will discuss the algorithm in detail and demonstrate how components of expected covariance can be obtained from the linked list that is the product of the algorithm.

Calculating an Expected Covariance Matrix using RAM

One general solution to covariance structural modeling involves constructing three matrices, a matrix \mathbf{A} of asymmetric relations, a matrix \mathbf{S} of symmetric relations, and a filter matrix \mathbf{F} that serves to distinguish latent from manifest variables (McDonald, 1978; McArdle & McDonald, 1984; McArdle & Boker, 1990). The \mathbf{A} matrix is of order $P \times P$ where P is the total number of variables and contains all regression weights expressed by the model such that element a_{ij} is the regression weight from variable j to variable i . Each non-zero element in the \mathbf{A} matrix corresponds to a single headed arrow in the corresponding path diagram.

The matrix \mathbf{S} is a symmetric $P \times P$ matrix that contains all variances and covariances in the model. An element s_{ij} represents the covariance between variable i and variable j . Similarly, an element s_{ii} represents the variance of the variable i . For each non-zero element in the lower triangle of the \mathbf{S} matrix there is a double headed arrow in the corresponding path diagram.

The filter matrix \mathbf{F} is a $m \times p$ matrix where p is the total number of variables and m is the number of manifest variables. The matrix \mathbf{F} has a 1 in each element f_{ij} such that variable i in the observed covariance matrix corresponds to the the variable j in the model.

The expected covariance matrix \mathbf{C}_e due to the model specified by \mathbf{A} , \mathbf{S} and \mathbf{F} can be

calculated as

$$\mathbf{C}_e = \mathbf{F}(\mathbf{I} - \mathbf{A})^{-1}\mathbf{S}((\mathbf{I} - \mathbf{A})^{-1})'\mathbf{F}' . \quad (1)$$

In turn, the matrix $(\mathbf{I} - \mathbf{A})^{-1}$ can be represented as a power series (McDonald, 1978)

$$(\mathbf{I} - \mathbf{A})^{-1} \approx \mathbf{A} + \mathbf{A}^2 + \mathbf{A}^3 + \dots . \quad (2)$$

A non-zero element of \mathbf{A} in row i and column j represents a path composed of one single headed arrow from variable j with variable i . A non-zero element of \mathbf{A}^2 in row i and column j contains the sum of all paths composed of the product of two single headed arrows leading from variable j through one mediating variable and then into variable i . In the same way, the matrix \mathbf{A}^3 contains elements that are the sum of paths of length 3 and so on. This critical relationship linking the product of connected paths with the powers of \mathbf{A} provides the means by which an automatic graph walking algorithm can be formed which restructures the covariance calculation into only its non-zero components.

By taking advantage of these relationships, we can thus construct a linked list that decomposes the expected covariance matrix into the individual paths that lead to the components of expected covariance between variables and also use this information as a set of constraints for automatically drawing path diagrams. This linked list of all paths of all lengths between all variables is also powerful diagnostic and didactic tool for understanding the structural relations within a structural equation model.

RAM Definition of an Example Model

We will define and step through the algorithm that creates the linked list of paths by operating on a simple example structural model. The model shown in Figure 3 will be used as an example. In this RAM style path diagram, every element has a specific meaning and exactly defines the model. Prior to creating the linked list of paths, we will define the example model and its diagram in terms of the RAM formulation.

The squares in the diagram represent manifest variables and the circles represent latent variables. Single headed arrows represent regression coefficients and double headed arrows represent covariances. The double headed arrows from a variable to itself represent the covariance between a variable and itself: its variance. In most path diagramming systems variances that are not residual variances are not drawn. In order to have a one to one mapping of all elements of the path diagram of a model to elements in the matrix formulation of the model RAM path diagrams impose the constraint that all non-zero arrows (single or double-headed) must be drawn.

Note that not all variables are directly connected by arrows. For instance, there is no single headed or double headed arrow directly between x_1 and y_1 . Thus the arrows that are shown in a diagram are drawn from a set of all possible arrows between any two variables. There are six variables in Figure 3, so there are $6 \times 6 = 36$ possible single headed arrows and $(6 \times 7)/2 = 21$ double headed arrows that could have been drawn. Any arrow that is not shown in the diagram still is possible, but if it is not drawn, it is constrained to be equal to zero. In this way, it is often more important to note what is *not shown* in a RAM path diagram than what *is shown*, since arrows not shown represent constraints.

If the variables in the model shown in Figure 3 are arranged in the ordered set $\{x_1, x_2, y_1, y_2, y_3, L\}$, the matrices which define this model in terms of the RAM formulation

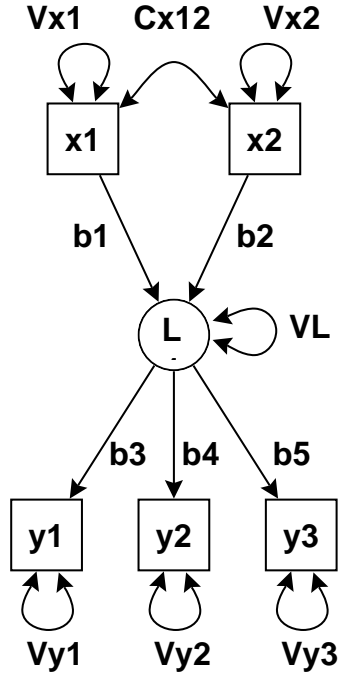


Figure 3. A simple example path model.

can be written as

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & b_3 \\ 0 & 0 & 0 & 0 & 0 & b_4 \\ 0 & 0 & 0 & 0 & 0 & b_5 \\ b_1 & b_2 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad (3)$$

$$\mathbf{S} = \begin{bmatrix} V_{x1} & C_{x12} & 0 & 0 & 0 & 0 \\ C_{x12} & V_{x2} & 0 & 0 & 0 & 0 \\ 0 & 0 & V_{y1} & 0 & 0 & 0 \\ 0 & 0 & 0 & V_{y2} & 0 & 0 \\ 0 & 0 & 0 & 0 & V_{y3} & 0 \\ 0 & 0 & 0 & 0 & 0 & V_L \end{bmatrix}, \text{ and} \quad (4)$$

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}. \quad (5)$$

Any possible single headed arrow between the variables in Figure 3 has a place in the $P \times P$ matrix \mathbf{A} . For instance the element $\mathbf{A}_{i,j}$ corresponds to the arrow pointing from

variable j to variable i in the ordered set of P variables. Thus, the regression coefficient $b3$ occupies the element $\mathbf{A}_{3,6}$ since $b3$ corresponds to the arrow pointing from L , the sixth member in the ordered set, and pointing to $y1$, the third member in the ordered set. It is therefore apparent that there is a one to one mapping between the 36 unique combinations of arrow beginnings and endings and the 36 cells in this matrix. While not all models that can be so described are sensible, all *possible* combinations of single headed arrows among P variables can be represented using this scheme for mapping between single headed arrows and elements in a $P \times P$ matrix \mathbf{A} .

Similarly, any possible double headed arrow in Figure 3 can be represented in the $P \times P$ symmetric matrix \mathbf{S} . A double headed arrow between members i and j in the ordered set of P variables can be thought of as pointing from and to both members i and j . Thus, the double headed arrow between i and j occupies both the element $\mathbf{S}_{i,j}$ and the element $\mathbf{S}_{j,i}$. Two example elements from the model shown in Figure 3 illustrate this mapping. There is a two headed arrow between $x1$ and $x2$ in the model representing the covariance $Cx12$ between $x1$ and $x2$. Since $x1$ and $x2$ are the first and second elements respectively in the ordered set of variables, the covariance between $x1$ and $x2$ is mapped to $\mathbf{S}_{1,2}$ and $\mathbf{S}_{2,1}$. Likewise, the double headed arrow $Vx1$ starting and ending at $x1$ corresponds to $\mathbf{S}_{1,1}$. Of course, when $i = j$, there is only one element in \mathbf{S} that is mapped to the double headed arrow since then $\mathbf{S}_{i,j} = \mathbf{S}_{j,i}$. Once again, although not all double headed arrows in a model are sensible, all possible combinations of double headed arrows between the variables in a length P ordered set can be mapped into the $P \times P$ matrix \mathbf{A} .

Finally, the $M \times P$ matrix \mathbf{F} is used to map the structural expectations of the full latent variable model given an ordered list of P total variables onto a second (possibly differently) ordered list of M manifest variables so that the covariance expectations of the model can be compared with observed covariances. The matrix \mathbf{F} consists of exactly M elements with value of one and all other elements have the value 0. The ones are arranged so that the rows and columns of the full latent variable model expectations are reordered to correspond to the order of the rows and columns of the observed covariances.

Although some might argue that this matrix formulation is arbitrary and contains a large proportion of zero elements, it has several desirable properties to recommend it. The first property is that this matrix formulation is entirely general. Thus, *any* single group linear structural model of covariance relations can be realized within its framework. What is and is not sensible is left entirely up to the modeler and is not imposed by constraints of the covariance expectation formula. Second, there is an exact one to one correspondence between this formulation and a diagrammatic representation. Thus, one may either draw a diagram or construct the RAM matrices and one has completely and unambiguously defined the other. Finally, taken together, this mapping between the diagram and the matrices when viewed in the context of the formula for the expected covariances leads to a straightforward and exhaustive method for calculating all expected individual components of expected covariance as well as an unambiguous method for displaying any arbitrary path diagram in a meaningfully simplified organization.

Tracing the Components of Covariance

When one assesses the simplicity of the path diagram in Figure 3 relative to the matrices \mathbf{A} and \mathbf{S} , it is apparent that there are many elements in matrices that are zero,

but the diagram represents these zero elements by simply not displaying anything. In this way, the directed graph (path diagram) of the model is an efficient descriptor in that it only represents elements that are non-zero. The algorithm presented here originally began as a method for quickly computing the inverse of the sparse matrix $(\mathbf{I} - \mathbf{A})$ using a directed graph representation. In the process of programming this calculation, it became apparent that one of the intermediary steps (an expanded linked list of paths) was useful in several ways other than that for which it was originally intended.

An expanded linked list of paths for a model is a data structure that allows one to quickly look up all of the connections from any one variable in a model to any other variable in a model. One particular type of connection is important when calculating the structural expectations of a model: a component of covariance between the two variables. Every component of covariance between two variables x and y follows the RAM tracing rule which consists of three parts, connected serially: (a) a path consisting of zero or more single headed arrows connected end to end and all pointing to the variable x , (b) a path consisting of zero or more single headed arrows connected end to end and all pointing to the variable y , (c) exactly one double headed arrow one end of which points to the base of path a and the other end of which points to the base of path b. For convenience, we have named such a component of covariance a *bridge*, a term from graph theory that describes such a connection between variables (variables are equivalent to *nodes* in graph theory). An example of this type of connection between x and y is shown in Figure 4.

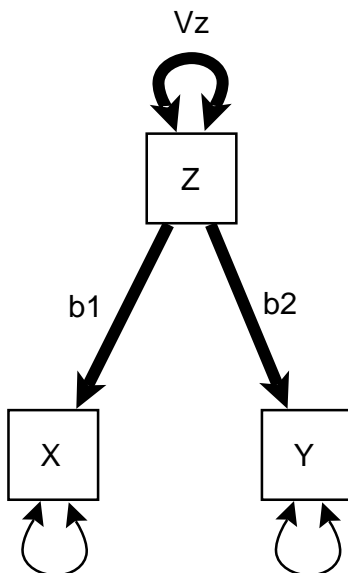


Figure 4. The component of covariance between x and y includes the path from z to x , the variance of z and the path from z to y . Thus the expected covariance between x and y is $C(xy) = b_1 V_z b_2$.

In our example model, finding the components of expected covariance between variables becomes more difficult. For instance, consider the components of expected covariance between y_1 and y_2 . There are five additive components to this covariance, each of which

follows the tracing rule defined above. These components are then summed to find the total covariance between y_1 and y_2 as illustrated in Figure 5. Finding the components by hand using the RAM tracing rule can be difficult and it is not always clear when one should stop looking for one more component. The expanded linked list of bridges provides an exhaustive list of all of the components of expected covariance between every pair of variables in a model.

The linked list of all components of expected covariance (bridges) is constructed in the following four steps:

1. Create a linked list of all single headed arrows in the model;
 2. Expand the linked list to all paths consisting of one or more single headed arrows;
- and
3. Create a list of all double headed arrows in the model;
 4. Combine the expanded linked list of paths with the list of double headed arrows to create the linked list of all bridges.

Creating and Expanding a Linked List of Paths

The first step in creating a linked list of all paths is to create a list of single headed arrows in the model. Given the model specified by the matrix \mathbf{A} , this is easily accomplished by simply listing the row, column and value (or symbol) associated with each non-zero element in \mathbf{A} . In order to later expand this list to include multi-arrow paths, several other columns will be created as shown in Table 1.

Table 1: A linked list of all single headed arrows of length 1 built from the \mathbf{A} matrix for the example model from Equation 3.

Index	From Variable	To Variable	Start Index	From Index	Length	Value
1	1	6	1	1	1	b_1
2	2	6	2	2	1	b_2
3	6	3	3	3	1	b_3
4	6	4	4	4	1	b_4
5	6	5	5	5	1	b_5

Each row in Table 1 has a unique integer assigned in the column labeled “Index”. The columns labeled “From Variable” and “To Variable” are the row and column numbers respectively from the \mathbf{A} matrix for the example model from Equation 3. The column labeled “Start Index” and “From Index” are for paths of length one simply the Index number from the current row. The column labeled “Length” holds the length of the current path and the column labeled “Value” holds the coefficient (symbolic or numeric) of the path, in this case taken from the \mathbf{A} matrix.

The next step is to expand the linked list of all single headed arrows to include all paths of length two. A path of length two must have included a variable which had a single headed arrow pointing to it as well as a single headed arrow pointing from it. Thus, we

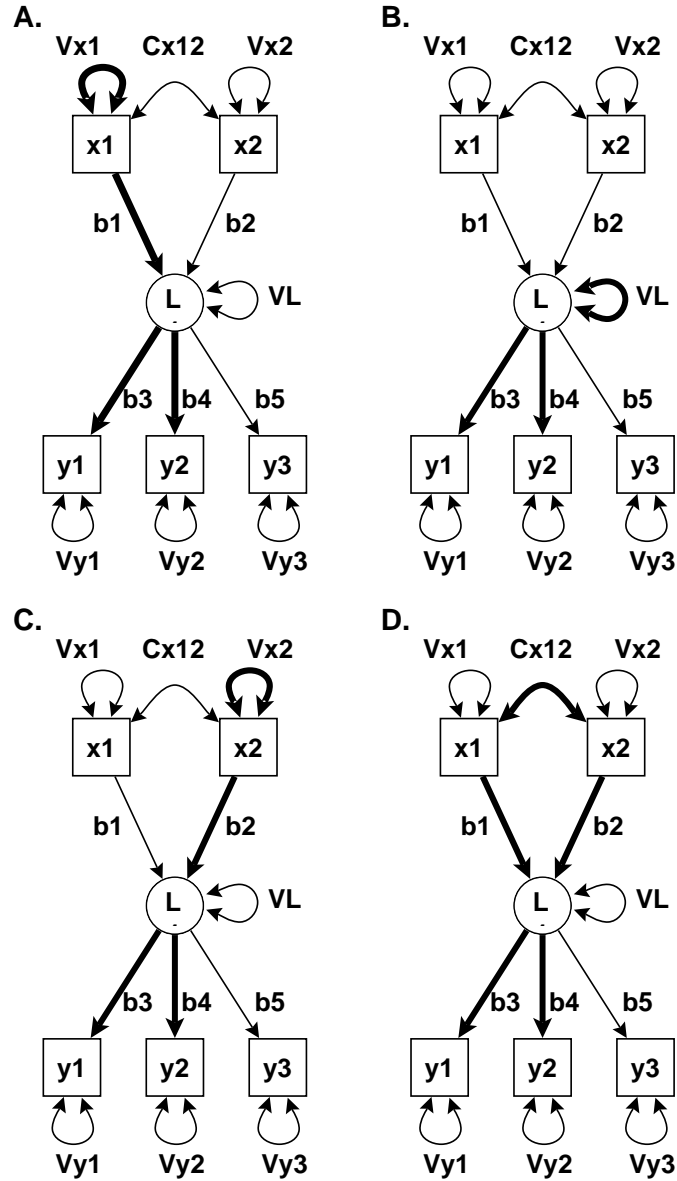


Figure 5. Components of covariance between y_1 and y_2 in the example model. There are five unique components of expected covariance which are summed to find the covariance between y_1 and y_2 $C(y_1, y_2) = b_3V_Lb_4 + b_3b_1V_{x1}b_1b_4 + b_3b_2V_{x2}b_2b_4 + b_3b_1C_{x12}b_2b_4 + b_3b_2C_{x12}b_1b_4$. Note that the components shown in the diagrams A, B, C are represented once, but the component in the diagram D can be constructed in two orders so it is represented twice.

can find an exhaustive list of paths of length two by performing the following steps for each path of length one.

1. Find the variable to which the selected path points.
2. Find all paths of length one which point from that variable.
3. For each combination of paths to and from the that variable, build an entry in the expanded list by extracting information from the list of paths of length one.

These steps are easily accomplished using the list of single headed arrows that has already been constructed. For instance, examine the path in Index position 1 from Table 1. This path is of Length 1, and its To Variable is 6. Thus for each time we find 6 in the From Variable column of paths of length 1 we construct a new entry in the table.

The first result of this process is shown in Index row 6 of Table 2. Index row 3 had a path of length 1 which has a From Variable value of 6 and To Variable value of 3. Thus in Index 6 we enter 1 into the From Variable and 3 into the To Variable, denoting that we are enumerating a path of length 2 that starts at variable 1 and ends at variable 3. The Start Index is set to 1, the Index value of the variable that starts this path. This information is kept since a single variable may have many arrows pointing from it, and thus just keeping the From Variable information is not enough to uniquely define the origin of the path. Next we enter 3 into the From Index column. This is the Index of the row where we just made the connection between the two paths of length 1. By saving this information, it is possible to walk back up a path, following where it came from step by step. This information turns out to be useful for automatic graphical placement, and may also prove to be useful in finding areas of local underidentification in a model. Finally, the value in Index 1 is multiplied with the value in Index 3 to provide a value for the path. These steps are repeated for each combination of paths of length 1 for which the To Variable of one of the paths matches the From Variable for another path.

Table 2: A linked list of all single headed arrows of lengths 1 and 2 built from the A matrix for the example model from Equation 3.

Index	From Variable	To Variable	Start Index	From Index	Length	Value
1	1	6	1	1	1	b_1
2	2	6	2	2	1	b_2
3	6	3	3	3	1	b_3
4	6	4	4	4	1	b_4
5	6	5	5	5	1	b_5
6	1	3	1	3	2	b_1b_3
7	1	4	1	4	2	b_1b_4
8	1	5	1	5	2	b_1b_5
9	2	3	2	3	2	b_2b_3
10	2	4	2	4	2	b_2b_4
11	2	5	2	5	2	b_2b_5

Paths of length three are constructed in a virtually identical manner. All paths of

length three must include both a path of length two pointing to some variable and a path of length one pointing from that same variable. Thus, by matching the values in the To Variable column for all paths of length two against the From Variable column for all paths of length one we can construct an exhaustive list of paths of length three. Examining the list in Table 2 reveals that the values 3, 4, and 5 appear in the To Variable column for paths of length two, but these values do not appear in the From Variable column of any path of length one. Thus there are no paths of length 3.

Paths of length $n + 1$ always must include a path of length n pointing to a variable and a path of length 1 pointing from that same variable. Since we have no paths of length 3, we can thus conclude that there can be no paths of length greater than three and the algorithm for expanding the list of single headed arrows has terminated. Since we now know that no paths of length greater than two exist and that paths of length one and of length two have been found, we can conclude that all paths of single headed arrows in the model are contained in the list in Table 2.

Termination with nonrecursive models

The type of approach used to identify and enumerate an exhaustive list paths is a recursive algorithm in that it relies on results from paths of length n to construct paths of length $n + 1$ and terminates by reasoning that if paths of length n do not exist, paths of length $n + 1$ cannot exist. However, so-called nonrecursive models have feedback loops in their paths and thus this recursive algorithm will not terminate using the rule proposed above. Nonrecursive models may be a bit of a misnomer in that they are so named since they are resistant to solution by recursive methods. However, in other fields, these models would be deemed to be infinitely recursive, since the reason they are resistant to solution by recursive algorithms is that the recursive algorithm does not terminate and continues to recurse indefinitely.

A rule can be added to the above algorithm which will always identify feedback loops of any length and terminate expansion of those feedback loops after a single circuit around the loop. Consider the path model shown in Figure 6. If we expand the list of paths for this model up to paths of length 3, it becomes immediately apparent what the rule should be.

First, we assign an ordering of the variables such that $X = 1$, $Y = 2$, $Z = 3$. Now the expanded list can be built and is shown in Table 3. Rows with Index values 7, 8 and 9 are the paths of length 3. Note that each of these rows, the From Variable is equal to the To Variable. Thus when a path both begins and ends at the same variable it is a feedback loop. This is a sensible and complete way of identifying all the paths associated with a feedback loop.

Thus we can construct a rule which will terminate in the presence of feedback loops of single headed arrows. Whenever a path of length n has a From Variable and To Variable that are equal, do not include that path in construction of paths of length $n + 1$. There remains a problem of the calculation of the value of any path that includes any portion of a feedback loop, since if the value of the minimum length path around a feedback loop is b , the value of all paths implicit in the the feedback loop is the power series

$$b + b^2 + b^3 + \dots \quad (6)$$

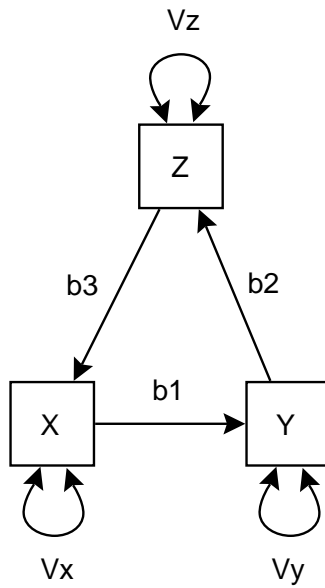


Figure 6. A “nonrecursive” path model.

Table 3: A linked list of all single headed arrows of lengths 1 and 2 built from the example model from Figure 6.

Index	From Variable	To Variable	Start Index	From Index	Length	Value
1	1	2	1	1	1	b_1
2	2	3	2	2	1	b_2
3	3	1	3	3	1	b_3
4	1	3	1	2	2	b_1b_2
5	2	1	2	3	2	b_2b_3
6	3	2	3	1	2	b_3b_1
7	1	1	1	4	3	$b_1b_2b_3$
8	2	2	2	5	3	$b_2b_3b_1$
9	3	3	3	6	3	$b_3b_1b_2$

for which there may or may not be a limit depending on the value of b .

More complicated nonrecursive models might include figure eight feedback loops. The stopping rule proposed above will correctly locate all of the unique combinations of smaller feedback loops that compose the interlocking feedback loops. Two loops with the same From Variable and To Variable endpoints will form an figure eight style of loop and the power series of these two loops will thus become combined.

Creating a List of Components of Covariance

Once we have created the expanded list of paths we can begin to build a list of components of expected covariance. Since every component of covariance includes exactly one double headed arrow, we first create a list of double headed arrows from the \mathbf{S} matrix. For each nonzero element in \mathbf{S} we create an entry in a list of double headed arrows as shown for the example model in Table 4. Note that we do not need some of the columns that had been used in the list of paths because the list of double head arrows only includes elements of length one. Also note that since the double headed arrows are not directional, the notions of “From” and “To” are not meaningful. Thus we have labeled the two columns “Variable A” and “Variable B” which contain the indices of the nonzero elements in the \mathbf{S} matrix.

Table 4: A linked list of all double headed arrows built from the \mathbf{S} matrix for the example model from Equation 4.

Index	Variable A	Variable B	Value
1	1	1	V_{x1}
2	2	2	V_{x2}
3	3	3	V_{y1}
4	4	4	V_{y2}
5	5	5	V_{y3}
6	6	6	V_L
7	2	1	C_{x12}
8	1	2	C_{x12}

Now, we are prepared to create a list of components of expected covariance for all pairs of variables in the model. Each component of covariance is a bridge, and must include exactly one double headed arrow and zero or one path starting from each end of the double headed arrow. Thus there are three configurations of bridges: (1) a double headed arrow alone, (2) a double headed arrow with a path from one of its ends, and (3) a double headed arrow with paths from both of its ends.

These three configurations of bridges can be built using the list of double headed arrows and the expanded list of paths. The following procedure will find all bridges.

1. List all of the double headed arrows.
2. List all pairs of one double headed arrow and one path such that either the “Variable A” or “Variable B” column for the double headed arrow matches the “From Variable” column in the expanded list of paths.

3. List all triplets consisting of one double headed arrow and two (not necessarily unique) paths such that the “Variable A” column for the double headed arrow matches the “From Variable” column for the first path “Variable B” column for the double headed arrow matches the “From Variable” column for the second path.

To help see how this works, a list of bridges that include the double headed arrow from X_1 to X_1 for the example model from Figure 3 has been calculated and listed in Table 5. It is apparent from the length of this list (25 separate components of covariance) that the number of bridges grows rapidly as the complexity of the model increases. However, a complete and exhaustive list of bridges can be automatically generated by computer using this algorithm, and it then becomes an easy matter to look up all of the possible components of covariance between any two selected variables. An Splus version of functions to produce a full list of bridges is provided on the web site <http://www.nd.edu/~sboker>. Using this software, the example model can be determined to have 119 bridges in all.

Table 5: A list of all bridges that include the double headed arrow from X_1 to X_1 in Figure 3.

Index	Variable A	Variable B	Span Index	Path 1 Index	Path 2 Index	Value
1	1	1	1	0	0	V_{x_1}
2	1	6	1	1	0	$V_{x_1}b_1$
3	1	3	1	6	0	$V_{x_1}b_1b_3$
4	1	4	1	7	0	$V_{x_1}b_1b_4$
5	1	5	1	8	0	$V_{x_1}b_1b_5$
6	6	1	1	0	1	$V_{x_1}b_1$
7	3	1	1	0	6	$V_{x_1}b_1b_3$
8	4	1	1	0	7	$V_{x_1}b_1b_4$
9	5	1	1	0	8	$V_{x_1}b_1b_5$
10	6	6	1	1	1	$b_1V_{x_1}b_1$
11	6	3	1	1	6	$b_1V_{x_1}b_1b_3$
12	6	4	1	1	7	$b_1V_{x_1}b_1b_4$
13	6	5	1	1	8	$b_1V_{x_1}b_1b_5$
14	3	6	1	6	1	$b_3b_1V_{x_1}b_1$
15	3	3	1	6	6	$b_3b_1V_{x_1}b_1b_3$
16	3	4	1	6	7	$b_3b_1V_{x_1}b_1b_4$
17	3	5	1	6	8	$b_3b_1V_{x_1}b_1b_5$
18	4	6	1	7	1	$b_4b_1V_{x_1}b_1$
19	4	3	1	7	6	$b_4b_1V_{x_1}b_1b_3$
20	4	4	1	7	7	$b_4b_1V_{x_1}b_1b_4$
21	4	5	1	7	8	$b_4b_1V_{x_1}b_1b_5$
22	5	6	1	8	1	$b_5b_1V_{x_1}b_1$
23	5	3	1	8	6	$b_5b_1V_{x_1}b_1b_3$
24	5	4	1	8	7	$b_5b_1V_{x_1}b_1b_4$
25	5	5	1	8	8	$b_5b_1V_{x_1}b_1b_5$

Using the Expanded Path List to Organize Graphics

If one compares Figures 1 and 2, it is immediately apparent that one diagram is more understandable than the other. However, both diagrams are equivalent with respect to their matrix formulation, and thus have the same expected covariance matrix. Clearly, the difference between these two diagrams does not lie in the algebra. Another way to say this is that the difference does not lie in the *topology* of these two diagrams, it lies in their *geometry*. The geometric placement of the variables on the page rather than the topologic nature of the connections between the variables is the reason these two diagrams differ.

Thus, arrangement of the variables on the page is a problem in sorting the variables. The relative distance between a chosen variable and all of the other variables in the diagram define where that variable is within the diagram. The expanded list of paths provides a way to sort variables so that they are in an ordering such that the distances between the variables correspond to geometric configurations of path diagrams as they are commonly presented. Consider Figure 7, where each variable is categorized according to the length of the longest path beginning at that variable.

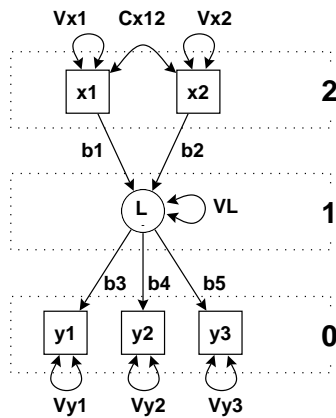


Figure 7. The example path model with variables categorized according to the length of the longest path beginning at each variable.

This categorization of variables into levels based on the length of the longest path which starts at that variable results in an automatic reduction in the apparent complexity of a path diagram. Just applying that one rule will reorganize Figure 1 into Figure 2. Similarly, by sorting the variables within a level so that the paths into those variables are in the same order as the order of the variables in the level above will change the path diagram of the confirmatory factor model shown in Figure 8–a into the path diagram in Figure 8–b.

Note that the variable Y_2 is predicted by both L_1 and L_2 . Variables that have more than one predictor from the level above can be sorted to occur at the appropriate boundary between variables with only one predictor. However, when there are more than two variables on the level above, a variable may have two predictors that are not contiguous at the upper level. In this case, the upper level should be sorted so that variables with two predictors can occur at the boundary between the variables with one predictor. An organization which

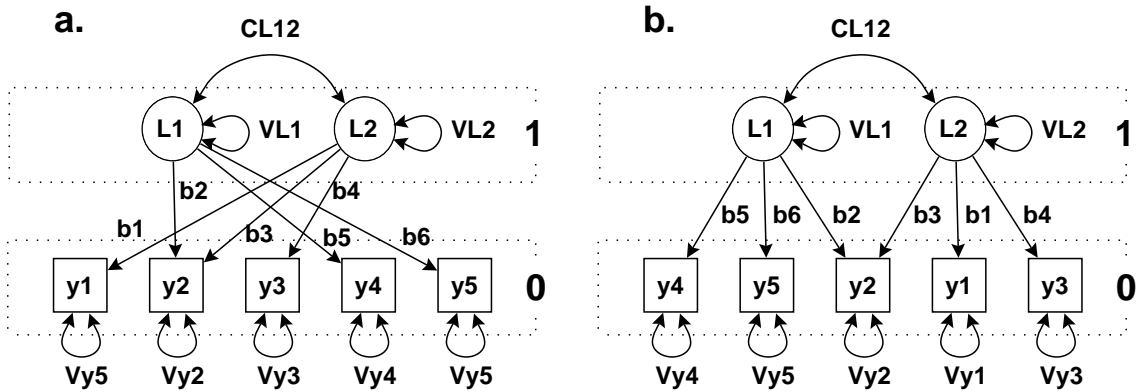


Figure 8. A confirmatory factor model with variables unsorted and sorted within levels. (a) The variables in level 0 are sorted by the order they appear in the A matrix. (b) The variables in level 0 are sorted by their inputs from the level above.

satisfies these constraints will not always be possible. For instance in Figure 9 there is one variable in level 0 that is predicted by L_1 and L_2 , one that is predicted by L_2 and L_3 , and one that is predicted by L_3 and L_1 . All arrangements of the factors will still produce a path diagram in which the single headed arrows cross.

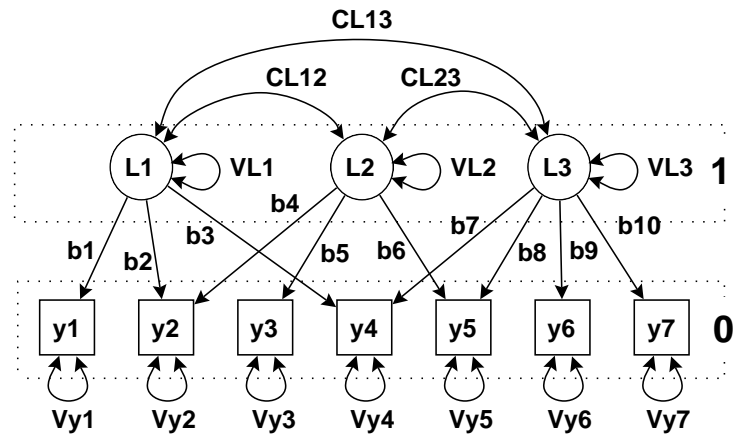


Figure 9. A confirmatory factor model with indicators that cannot completely satisfy the level sorting constraints.

Finally, path models that contain feedback loop paths need to be addressed. We recommend placing feedback loops as a unit within the level defined by the length of the longest path beginning at a variable in the feedback loop and containing no other variables from within the feedback loop. Thus, the nonrecursive portion of a model can be drawn so that, as much as possible, the elements of a feedback loop are nearest neighbors to each other. This is sensible when one considers that the value of a feedback loop is a power series

of the linear combination of the elements contained within the feedback loop.

Examining Local Structure

Although we have just spent a good deal of time discussing what might only appear to be aesthetics of diagrams, the automatic geometric layout of a path diagram can be informative about the form of simple structure that is implied by a theory. As structural modelers, we are concerned with constructing models that attain a good fit to data relative to their simplicity. Most goodness of fit measures are, in one way or another, concerned with this question. But simplicity can be attained in many ways. Most measures of model simplicity are only concerned with an overall count of parameters and statistics.

However, it is often the case that simplicity in a model varies across the model; in other words some of the variables can contribute more to the misfit of the model than others. In some cases, a portion of a model may be exactly identified, or underidentified even though it appears that there are enough degrees of freedom overall. One may use the automatically generated sorting of the path model to examine the local structure of a model. Here, the notion of locality is refined by the sorting that has occurred in order to define geometric distance between variables so that an organized path diagram could be drawn.

In order for a latent variable to be scaled, at least one path leading from it or its variance must be fixed. This is the commonly used criterion for making sure that a latent variable is identified. However, many times modelers will ignore the fact that a latent variable must have three indicators before that latent construct is structurally identified. In fact, a latent variable with only three indicators is locally exactly identified, that is, there is no opportunity for that local neighborhood to contribute to the misfit of the model. A latent construct needs at least four indicators before this portion of the model is overidentified.

Neighborhood categories can be defined by organizing the expanded list of paths into groups of paths that all share the same origin. Each of these neighborhoods can then be assessed for sufficiency of identification. Neighborhoods may or may not overlap, and one may wish to analyze highly overlapping neighborhoods as if they were one neighborhood.

Discussion

Structural equation models can and are frequently represented as path diagrams. We have explored one method for using the one-to-one relationship between RAM path diagrams and their structural expectations to examine the additive components that define the structural expectations inherent in a model, what we have called bridges. These components of covariance expectation are in turn composed of paths and spans. By examining paths, we can automatically generate path diagrams from their algebraic descriptions using a rule that categorizes a variable based on the longest path beginning at the chosen variable. This rule produces path diagrams that are similar to path diagrams that are commonly drawn.

The methods presented here are especially interesting in that they provide a logical way to evaluate the structural sensibility of models. Models that have simple structure are generally preferred over models with more complex structure (Ressler, 1963; Thurstone, 1938, 1947). The methods presented here could be used to provide objective criteria for evaluating what is meant by “simple structure”.

We have presented an overview of how the expanded lists of paths and spans can be used to define locality and neighborhoods in a model in such a way that issues of local identification can be addressed. In all, these methods for analyzing a structural equation model provide detailed and useful information that is otherwise not available.

Finally, we hope that these tools can be used to gain more complete understanding of the implications of the addition or deletion of particular paths in a model. Does the addition or deletion of a path mean that a variable has changed levels in the hierarchy of variables? If so, a structural change has been made that involves a greater difference in complexity than one that does not force such a change in levels. Does the addition or deletion of a path force the resorting of variables within a level? If so, a change has been made that influences the degree of overlap of neighborhoods and so has changed the complexity of a model to a greater degree than the addition or deletion of a path which does not force such a reorganization.

Conclusion

The specification of a structural model in RAM terms is general and complete. However, other more restrictive algebraic formulations are often used since the models so specified are more likely to be sensible given their framework of analysis. However, restricting oneself to what is sensible today often imposes unnecessary restrictions on new methods tomorrow. We have discussed a set of methods which allow one to evaluate the structural sensibility of a model without placing hard structural constraints within the formulation of the expectation algebra. In this way, we hope to encourage more flexible thinking about the way that structural equation models may be specified and interpreted.

References

- Arbuckle, J. L. (1997). *Amos user's guide. version 3.6*. Chicago: SPSS.
- Bentler, P. M. (1995). *EQS structural equations program manual*. Encino, CA: Multivariate Software.
- Christofides, N. (1975). *The theory and application of graphs*. New York: Academic Press.
- Gemignani, M. C. (1972). *Elementary topology*. Reading, MA: Addison–Wesley.
- Jöreskog, K. G., & Sörbom, D. (1996). *LISREL 8: A guide to the program and applications (2nd ed.)*. Chicago: Scientific Software International.
- McArdle, J. J., & Boker, S. M. (1990). *Rampath*. Hillsdale, NJ: Lawrence Erlbaum.
- McArdle, J. J., & McDonald, R. P. (1984). Some algebraic properties of the Reticular Action Model for moment structures. *British Journal of Mathematical and Statistical Psychology*, 87, 234–251.
- McDonald, R. P. (1978). A simple comprehensive model for the analysis of covariance structures. *British Journal of Mathematical and Statistical Psychology*, 31, 59–72.
- Muthén, B. O., L. K. & Muthén. (1998). *Mplus user's guide*. Los Angeles: Muthén & Muthén.
- Neale, M. C., Boker, S. M., Xie, G., & Maes, H. H. (1999). *Mx: Statistical modeling*. (Box 126 MCV, Richmond, VA 23298: Department of Psychiatry, 5th Edition)
- Ressler, R. H. (1963). Some justifications for parsimony in psychology. *Psychological Reports*, 13(3), 643–646.

Thurstone, L. L. (1938). The current misuse of the factorial methods. *Psychometrika*, 2, 73–76.

Thurstone, L. L. (1947). *Multiple factor analysis*. Chicago: University of Chicago Press.