Unbelievably Fast Estimation of Nested Multilevel Structural Equation Models

Joshua Nathaniel Pritikin
Charlottesville, VA

M.A. Psychology, University of Virginia, 2013
B.S. Psychology, University of Oregon, 2009

A Dissertation presented to the Graduate Faculty
of the University of Virginia in Candidacy for the Degree of
Doctor of Philosophy

Department of Psychology

University of Virginia
April, 2016

Abstract

We introduce relational SEM, an adaptation of structural equation modeling to relational databases. Relational SEM is a superset of the mixed model and multilevel SEM. In addition, we introduce Rampart, a new computational strategy for frequently encountered relational SEM models with all continuous indicators. Rampart is inspired by the fact that the multivariate normal density is transparent to orthogonal rotation. Well suited to big data, Rampart becomes more effective as the size of the data set increases. When data are strictly nested then there are usually fewer variables in the upper level connected to many more variables in the lower levels. A regression from teacher skill to student performance has this characteristic. In such a model, under typical conditions, a rotation can be applied to eliminate all but one of the links from teacher to student with a corresponding rotation applied to the observations. This transformation leaves the likelihood function unchanged, but offers a major benefit: dramatically increased independence in the model implied covariance matrix. Rampart requires strictly nested structure and identical sub-models. Rampart can be applied locally to the part of a model that meets these criteria. Rampart is implemented in `OpenMx`. `OpenMx` is free and open software that runs on all major operating systems.

Contents

Introduction

Many non-statisticians have an intuitive notion of variability of a indicator and association between two indicators. We cannot entertain causal theories without these notions. When an infant learns that crying will cause her parents to offer her water, food, and a diaper change, these statistical engines are probably at work. Not all processes are best described by a Gaussian distribution. However, the non-Gaussian part is often confined to the outer vertices of a casual graph while the central part of the graph remains Gaussian. The Gaussian distribution is of central importance in statistics and causal reasoning (Pearl, 2000; Voelkle & Oud, 2013).

Gaussian Models

Let parameter vector $\boldsymbol{\theta} \equiv \{\boldsymbol{\mu}, \boldsymbol{\Sigma}\}$ with $\boldsymbol{\mu}$ as a K dimensional mean vector (1st moment) and $\boldsymbol{\Sigma}$ as a $K \times K$ covariance matrix (2nd moment). For data $\boldsymbol{y}$ and with some regularity assumptions, the Gaussian log density can be written as,

$$\ell(\boldsymbol{y}|\boldsymbol{\theta}) = \sum_i \left[-\frac{1}{2} \left[K \log(2\pi) + \log(|\boldsymbol{\Sigma}|) \right] - \frac{1}{2} (\boldsymbol{\mu} - \boldsymbol{y}_i)^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{\mu} - \boldsymbol{y}_i) \right]. \quad (1)$$

It is no overstatement to say that this model has a rich history in the annals of statistics.

Similar to the way that some countries that were slow to implement a wired phone system have skipped directly to wireless phones, we are now at a stage of Gaussian model development where great swaths of less productive detours can be skipped. The history of the Gaussian model has grown sprawling and convoluted. Diverse special purpose models once conceived independently can now be re-expressed as variations of a general model. We introduce the general model with a judicious review of the essential building blocks.

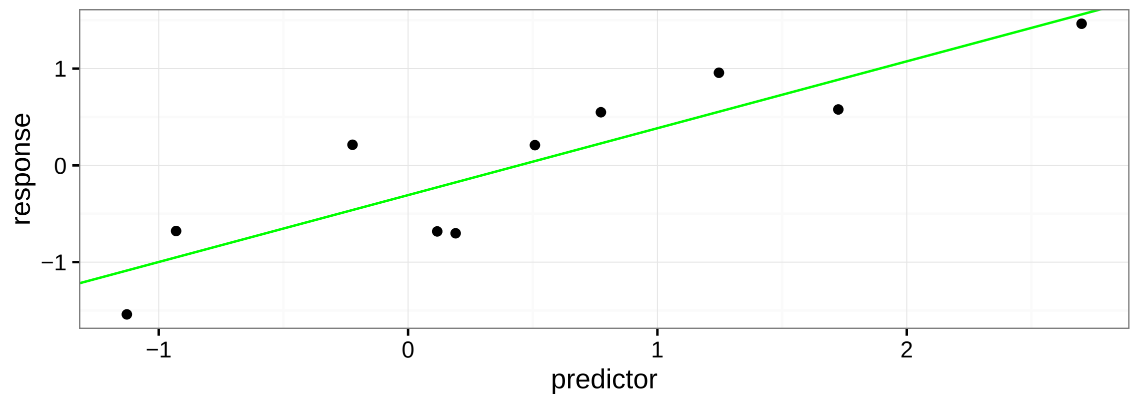

Figure 1. Data are shown as points with the least squared residual regression line.

Table 1
Example data for linear regression.

	predictor	response
	0.12	−0.68
	1.73	0.58
	1.25	0.96
	0.51	0.21
	−1.13	−1.54
	−0.93	−0.68
	0.19	−0.70
	−0.22	0.21
	2.70	1.46
	0.77	0.55
$\mu =$	0.5	0.04
$\sigma =$	1.18	0.92

Linear Regression

In the 1870s, Galton and colleagues devised linear regression (Stanton, 2001). Linear regression answers questions of the form, given n independent measurements of predictor x and response y, what approximation to

$$\boldsymbol{y} = \alpha + \beta \boldsymbol{x} \tag{2}$$

minimizes the squared residual.[1] The solution is

$$\beta = \frac{\mathrm{Cov}(\boldsymbol{x}, \boldsymbol{y})}{\mathrm{Var}(\boldsymbol{x})} \tag{3}$$

$$\alpha = \bar{x} - \bar{y}\beta. \tag{4}$$

For example, given data in Table 1 ($n = 10$),

$$\beta = \frac{0.96}{1.38} = 0.69 \tag{5}$$

$$\alpha = 0.04 + (-0.5)\beta = -0.31. \tag{6}$$

The data and regression line are plotted in Figure 1.

Developed in the olden days before computers, regression was originally framed in terms of squared residuals because computational simplicity was the overriding concern. The modern day statistical engine, Bayes' Theorem (Equation 16), had been disseminated in 1763 but would not blossom until Fisher conceived the method of maximum likelihood in the 1920s. Fortuitously, if we specify a Gaussian model for the data and assume that the residual is independently, identically, and normally distributed then the least squared residual criterion identifies the the same estimates as would be found using Fisher's modern maximum likelihood approach.

Analysis of Variance (ANOVA)

Analysis of variance is concerned with detection of group differences. The simplest version was formally introduced by Fisher in the 1920s. Like linear regression, ANOVA was originally framed in terms of squared differences instead of in terms of Bayes' Theorem. Suppose we want to determine if two groups are different on some

[1] We use the term *residual* instead of *error* because the connotations of *error* are not always appropriate.

Table 2
Example data for one-way analysis of variance with groups 1 and 2 in columns.

	1	2
	1.18	−0.52
	0.32	0.86
	0.88	1.29
	1.46	1.13
	−0.31	−0.30
	−0.91	1.37
	0.42	−0.15
	0.14	1.89
	−0.17	0.69
	−0.06	−0.32
$\mu =$	0.3	0.6
$\sigma =$	0.72	0.85

measure y. An F statistic can be obtained with,

$$SS_{between} = \sum_{j=1}^{2} (\bar{y}_j - \bar{y})^2 \tag{7}$$

$$SS_{within} = \sum_{j=1}^{2} \sum_{i=1}^{n_j} (y_{ij} - \bar{y}_j)^2 \tag{8}$$

$$F = \frac{SS_{between}/1}{SS_{within}/(N-2)}. \tag{9}$$

For example, given the data in Table 2,

$$SS_{between} = 0.44 \tag{10}$$

$$SS_{within} = 11.22 \tag{11}$$

$$F = \frac{0.44}{0.62} = 0.71. \tag{12}$$

While convenient for hand calculation, the method framed in terms of squared differences obscures the relationship between ANOVA and linear regression. The two models are almost the same (compare with Equation 2) except that here x is a binary indicator of group membership,

$$\boldsymbol{y} = \alpha + \beta\boldsymbol{x}. \tag{13}$$

If we code group 2 as $x = 1$ then

$$\alpha = \bar{y}_1 = 0.3 \tag{14}$$

$$\beta = \bar{y}_1 - \bar{y}_2 = 0.3 \tag{15}$$

The t value for the null hypothesis that $\beta = 0$ is not such a simple calculation, but it can be obtained with R to cross-check the magnitude of $\sqrt{F} = 0.84$

```
summary(lm(y~group, aovData))$coefficients['group2','t value']

## [1] 0.84
```

The Mixed Model

Linear regression and ANOVA models introduce two different kinds of coefficients. In linear regression (Equation 2), β helps predict every observation whereas in ANOVA (Equation 13), β only helps predict a subset of observations. This is an important distinction. Historically, coefficients that help predict all observations are called *fixed effects* whereas the other type of coefficient has been called a *random effect*. These are an unfortunate terminology. In the statistical literature, there are at least five definitions of these phrases, all of which differ from each other (Gelman, 2005). Moreover, in computer science, the term *random* is usually associated with draws from a uniform random number generator, not synonymous with *stochastic* that does not suppose a particular distribution. Here we follow Gelman (2005) and use the terms *constant* and *varying*. For example, the model $y_{ij} = \alpha_j + \beta x_{ij}$ has varying intercepts α_j and a constant slope β. Models with both kinds of coefficients, constant and varying, are called *mixed* models.

As foreshadowed, the squared residuals or squared differences approach to model estimation imposes inconvenient restrictions. To perform ANOVA using squared differences, all combinations of conditions must have an equal number of samples and there is no simple way to cope with missing data. There are some ways to finesse the problem (e.g., Henderson, 1953), but a much more robust solution is to embrace Bayes' Theorem. Let $\boldsymbol{\theta}$ be a vector of model parameters. Bayes' Theorem is,

$$\Pr(\boldsymbol{\theta}|data) = \frac{\Pr(data|\boldsymbol{\theta})\Pr(\boldsymbol{\theta})}{\Pr(data)}. \tag{16}$$

Since $\Pr(data)$ does not depend on the parameters $\boldsymbol{\theta}$, we can omit it, leaving

$$\Pr(\boldsymbol{\theta}|data) \propto \Pr(data|\boldsymbol{\theta})\Pr(\boldsymbol{\theta}). \tag{17}$$

This equation is of such paramount importance that special names are assigned to each term. The density $\Pr(\boldsymbol{\theta})$ is the *prior*, $\Pr(data|\boldsymbol{\theta})$ is the *likelihood*, and $\Pr(\boldsymbol{\theta}|data)$ is the *posterior*.[2] For even modestly complex models, the posterior $\Pr(\boldsymbol{\theta}|data)$ can be impractical to understand directly. To explore and summarize the posterior, at least two popular approaches are available. One approach is to sample from the posterior, typically using some kind of Markov-Chain Monte Carlo (MCMC) method (e.g., Plummer, 2013; Stan Development Team, 2014). From these samples, mean point estimates and their marginal distributions can be obtained. The second approach is to treat the likelihood or posterior as an arbitrary function and find its mode. This method was introduced by Fisher in the 1920s under the name *maximum likelihood* (Efron, 1998). Some controversy surrounds the prior $\Pr(\boldsymbol{\theta})$ (e.g., Gelman, 2008),

[2]Likelihood is not synonymous with probability. Consider P(A|B), a function of both A and B. For fixed B, P(A|B) is the probability of A conditional on B. For fixed A, P(A|B) is the likelihood of B (MacKay, 2003, p. 28).

but we have no qualms about it and consider *maximum likelihood* synonymous with *maximum posterior*.

Different ways of summarizing the posterior have strengths and weaknesses. The MCMC approach can obtain posterior means that are more stable than posterior modes when the posterior has multiple peaks of nearly equal height. However, unresolved questions remain about how to infer MCMC convergence (Gelman & Shirley, 2011). The present article focuses on the mode instead of mean.

A desire for models with arbitrary combinations of constant and varying coefficients without onerous restrictions on data structure culminated in a maximum likelihood estimation method for the mixed model (Hartley & Rao, 1967). For a column vector of observations $\boldsymbol{Y}$, covariates $\boldsymbol{X}$ associated with constant coefficients $\boldsymbol{\beta}$, covariates $\boldsymbol{Z}$ associated with varying coefficients $\boldsymbol{u}$, and a column vector of residuals $\boldsymbol{e}$, the mixed model can be written as,

$$\boldsymbol{Y} = \underbrace{\boldsymbol{X\beta}}_{\text{constant}} + \underbrace{\boldsymbol{Zu} + \boldsymbol{e}}_{\text{varying}}. \tag{18}$$

To better appreciate the flexibility of this model, we suspend our presentation here without discussion of the distributional assumptions. A mixed model is often specified as a regression formula. A weakness of regression formulae are that they only specify the model for the first moment ($\boldsymbol{\mu}$ of Equation 1). Specification of the second moment ($\boldsymbol{\Sigma}$ of Equation 1) is assumed as a well known default. As an alternative, both moments of a model can be specified simultaneously using a path diagram.

Path Diagrams

In the 1970s, two different Gaussian model specification languages emerged, LISREL (Jöreskog & Van Thillo, 1972) and COSAN (McDonald, 1978). In the process of reconciling these two different specifications, the Reticular Action Model (RAM) was distilled (McArdle, 2005; McArdle & McDonald, 1984). Although LISREL, COSAN,

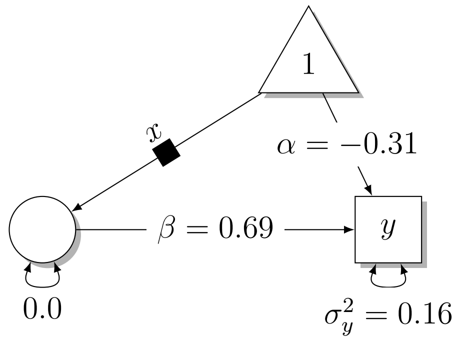

Figure 2. Equation 2 drawn as a RAM path diagram. The triangle acts like an observed variable that is always 1. The square and circle denote observed and latent variables, respectively. The black square on the path from the triangle to the circle is a definition variable. Single-headed arrows are regressions and double-headed arrows are variances. The diagram takes up more space on the page compared to Equation 2, but it also makes the covariance model explicit. The variance for x is not estimated. σ_y^2 is regarded as the residual variance.

and RAM offer equivalent expressive power, the RAM model is the most parsimonious of the three. Moreover, there is a one-to-one correspondence between the RAM model and intuitive path diagrams. In contrast to regression formulae, RAM path diagrams incorporate specification of both the first and second moments.

The RAM model consists of 4 matrices, traditionally called $\boldsymbol{A}$ (asymmetric), $\boldsymbol{S}$ (symmetric), $\boldsymbol{F}$ (filter), and $\boldsymbol{M}$ (mean). The RAM matrices are related to the model's Gaussian distribution by,

$$\boldsymbol{\mu} = \boldsymbol{F}(\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{M} \tag{19}$$

$$\boldsymbol{\Sigma} = \boldsymbol{F}(\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{S}(\boldsymbol{I} - \boldsymbol{A})^{-T}\boldsymbol{F}^{T}. \tag{20}$$

These equations may appear daunting, but note that when $\boldsymbol{A}$ is zero and $\boldsymbol{F}$ is the identity matrix then $\boldsymbol{\mu} = \boldsymbol{M}$ and $\boldsymbol{\Sigma} = \boldsymbol{S}$. So what is the purpose of $\boldsymbol{A}$ and $\boldsymbol{F}$? The $\boldsymbol{A}$ matrix comes into play in the specification of regression relationships. Our linear regression (Equation 2) can be diagrammed as in Figure 2. The multivariate generalization of Equation 4 is implemented by the products that involve $(\boldsymbol{I} - \boldsymbol{A})^{-1}$.

Table 3
Example data for latent factor model.

	x1	x2	x3
	−0.99	−0.79	−0.67
	0.05	−2.48	−0.64
	−1.30	−0.82	−1.06
	−1.49	−1.76	−1.28
	1.14	1.18	1.06
	0.96	0.62	0.91
	−0.26	−0.17	−0.25
	−0.83	1.33	−0.00
$\mu =$	−0.34	−0.36	−0.24
$\sigma =$	1	1.37	0.86

The $\boldsymbol{F}$ matrix is used to filter out variables from the model, permitting these variables to be latent (not measured). Latent variables were devised by Spearman in the early 1900s (P. Lovie & A. D. Lovie, 1996). For example, Figure 3 exhibits a latent factor model with 3 observed indicators. To clarify how this model works, the corresponding RAM matrices are given along with the model expected covariance $\boldsymbol{\Sigma}$,

$$\boldsymbol{F} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{21}$$

$$\boldsymbol{A} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & \lambda_{x2} \\ 0 & 0 & 0 & \lambda_{x3} \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{22}$$

$$\boldsymbol{S} = \begin{pmatrix} \sigma^2_{x1} & 0 & 0 & 0 \\ 0 & \sigma^2_{x2} & 0 & 0 \\ 0 & 0 & \sigma^2_{x3} & 0 \\ 0 & 0 & 0 & \sigma^2_g \end{pmatrix} \tag{23}$$

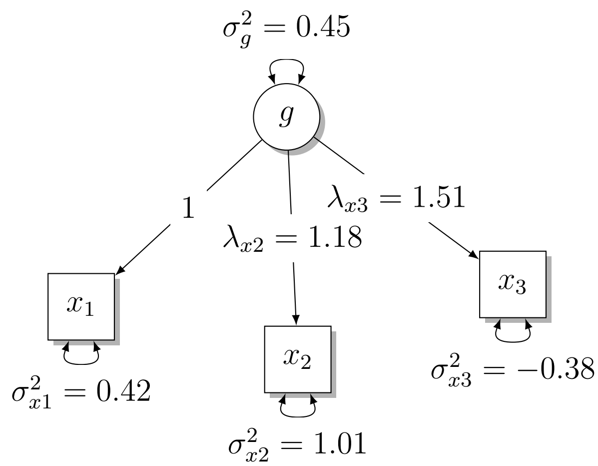

Figure 3. A latent factor model given the observed data in Table 3. The latent factor is drawn with a circle. The regression from g to x_1 has a fixed loading of 1. Note that σ^2_{x1}, σ^2_{x2}, and σ^2_{x3} are unique factor variances.

$$\boldsymbol{\Sigma} = \boldsymbol{F}(\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{S}(\boldsymbol{I} - \boldsymbol{A})^{-T}\boldsymbol{F}^T = \begin{pmatrix} \sigma^2_g + \sigma^2_{x1} & \sigma^2_g\lambda_{x2} & \sigma^2_g\lambda_{x3} \\ \sigma^2_g\lambda_{x2} & \sigma^2_g\lambda^2_{x2} + \sigma^2_{x2} & \sigma^2_g\lambda_{x2}\lambda_{x3} \\ \sigma^2_g\lambda_{x3} & \sigma^2_g\lambda_{x2}\lambda_{x3} & \sigma^2_g\lambda^2_{x3} + \sigma^2_{x3} \end{pmatrix}. \tag{24}$$

There are 6 parameters. Since the observed covariance matrix has 6 non-redundant entries, this model is just specified. In modeling, latent factors can be treated as if they represent regular observed scores. If factor scores are desired then various ways are available to estimate them (e.g., Estabrook & Neale, 2013) as long as identifying assumptions are made. In summary, latent factors are an ingenious user interface. Without the RAM parameterization, it would be more difficult to learn how to specify Equation 24.

A Gaussian parameterization that is well suited for estimation of latent factors and regressions is often called a *structural equation model* (SEM; Fan, 1997). We regard RAM as an SEM parameterization of the Gaussian model. To review, using RAM we can specify constant coefficients (1st and 2nd moment) in covariance or regression form with respect to observed variables or latent factors. Originally, RAM

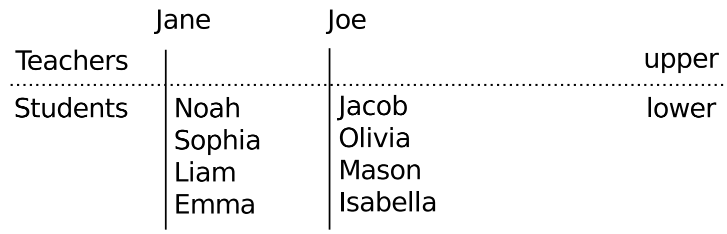

Figure 4. Students nested within teachers. For example, Noah is Jane's student and Jacob is Joe's student. There is a one-to-many relationship between teachers and students. A different model would be needed to accommodate students that spent some proportion of their time with each teacher.

did not provide any special support for varying coefficients. Recently, at least one proposal to extend RAM path diagrams to arbitrarily varying coefficients has been advanced (Curran & Bauer, 2007). Circles, traditionally used to represent latent factors, were re-purposed to represent varying coefficients. This makes sense because varying coefficients are a more general concept than latent factors. A latent factor is equivalent to a coefficient varying by individual with constant estimated loadings to indicators. At this stage, it may be difficult to judge the merit of Curran and Bauer's proposal due to the potential diverse uses of varying coefficients. To better focus our user interface concerns, we introduce a major application of varying coefficients: multilevel structure.

Multilevel structure

The simple aggregation of observations (Equation 1) is contingent on the assumptions that observations are independent and identically distributed. For example, students within a single classroom may exhibit independent performance. However, students drawn from two different classrooms may exhibit some classroom specific effect. Across classrooms, we can no longer consider the individual student as an independent unit of analysis (Kenny & Judd, 1986).

Data with complex structure are often stored in relational databases. Typically, data are normalized into *first normal form*, eliminating redundant or repeating data.

Primary keys are assigned to uniquely identify entities. Foreign keys refer to primary keys, allowing the relationships between the data tables to be recovered by the join of primary and foreign keys (e.g., Maier, 1983). Data are considered multilevel when an independent unit of analysis must span across two or more normalized database tables. For example, data on students and teachers would be stored in at least two tables. These data must be stored in separate normalized tables because there is not a 1-to-1 relationship between students and teachers. Since there are fewer teachers than students, teachers are regarded as the *upper* level and students as the *lower* level (see Figure 4).

To describe model structure when there are more than 2 levels we need to introduce two more terms, *nested* and *crossed*. Data are *nested* when each lower level partition is contained within its upper level. When data are not nested then they are *crossed*. Crossed varying coefficients need not be organized in relation to other varying coefficients. Crossed coefficients may partition observations in arbitrary ways. For example, suppose a school reassigns some of its students to different classrooms halfway through the year. If we study the whole year, some students will have single teachers but some will have two teachers. Students with two teachers involve a crossed assignment of varying coefficients. The distinction between nested and crossed data is useful because nested data are easier to process than crossed data.

Modeling multilevel data is one of the major applications of varying coefficients. Suppose the focus of our analysis is students. We want to estimate a few constant regression coefficients to learn how student performance depends on socioeconomic status and some intervention. We would like to specify our relationships in terms of latent factors because we cannot measure any of the constructs of interest directly. However, we need to incorporate varying coefficients in the model to properly account for teacher effects within a school, school effects within a district, and district effects within a state. If we proceed along these lines, the independent units of analysis are the highest level units, perhaps entire states.

The bottleneck in the evaluation of Equation 1 is the matrix inverse of the model implied covariance matrix $\mathbf{\Sigma}$. Gauss-Jordan matrix inverse requires $O(n^3)$ operations. To fit multilevel models quickly, it is essential to analyze the structure of this matrix and devise some way to reduce its dimension or complexity. Before we discuss optimization techniques, it will be helpful to sketch out more concretely the structure of our hypothetical multilevel student model covariance matrix. To keep things simple, assume the data are nested (not crossed). We introduce the *direct sum* operator,

$$\boldsymbol{B}_1 \bigoplus \boldsymbol{B}_2 = \begin{pmatrix} \boldsymbol{B}_1 & \mathbf{0} \\ \mathbf{0} & \boldsymbol{B}_2 \end{pmatrix}$$

$$\bigoplus_{i=1}^{k} \boldsymbol{B}_i = \begin{pmatrix} \boldsymbol{B}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{B}_2 & & \vdots \\ \vdots & & \ddots & \mathbf{0} \\ \mathbf{0} & \cdots & \mathbf{0} & \boldsymbol{B}_k \end{pmatrix}$$

to conveniently construct these matrices. Suppose we build a covariance model $\boldsymbol{S}$ for a particular student. A classroom of s students will have covariance matrix

$$\boldsymbol{T} = \begin{pmatrix} \boldsymbol{T}_{1,1} & \boldsymbol{T}_{1,2} \\ \boldsymbol{T}_{2,1} & \bigoplus_{i=1}^{s} \boldsymbol{S}_i \end{pmatrix}. \tag{25}$$

That is, each student is independent of other students, $\boldsymbol{T}_{1,1}$ is square, and $\boldsymbol{T}_{1,2}$ and $\boldsymbol{T}_{2,1}$ are rectangular. The quadrants labeled with $\boldsymbol{T}$ represent classroom or teacher relationships with each student. This pattern continues as we move up levels. A

Employee	Dept
Harry	Sales
Sally	Finance
George	Finance
Harriet	Sales

Dept	Manager
Sales	George
Finance	Harriet
Production	Charles

Employee ⋈(Dept) Manager

Employee	Dept	Manager
Harry	Sales	George
Sally	Finance	Harriet
George	Sales	George
Harriet	Finance	Harriet

Figure 5. An employee table (a.k.a relation or data frame) and manager table are given (upper tables). The employee and manager tables are joined by department (lower table).

school of t classrooms will have covariance matrix

$$\boldsymbol{H} = \begin{pmatrix} \boldsymbol{H}_{1,1} & \boldsymbol{H}_{1,2} \\ \\ \boldsymbol{H}_{2,1} & \bigoplus_{i=1}^{t} \boldsymbol{T}_i \end{pmatrix} \tag{26}$$

and a district of h schools will have covariance matrix

$$\boldsymbol{D} = \begin{pmatrix} \boldsymbol{D}_{1,1} & \boldsymbol{D}_{1,2} \\ \\ \boldsymbol{D}_{2,1} & \bigoplus_{i=1}^{h} \boldsymbol{H}_i \end{pmatrix}. \tag{27}$$

Without working out the exact shape of such a covariance matrix, it should be clear that it can be very large and very sparse.

Relational algebra

Before we proceed to other topics, this is a good point to formally describe how data is combined in multilevel models and the corresponding `OpenMx` user interface. Let R and S be tables (or data frames) that contain rows. A row is a single unit of data like the data for one teacher or one student. Following standard relational database theory (e.g., Maier, 1983), the join operator ($\bowtie$) is defined as,

$$R \bowtie(F) \; S \equiv \{r \cup s \wedge r \in R \wedge s \in S \wedge F(r \cup s)\}$$

where F is a boolean valued function. Without loss of generality, here F tests whether primary and foreign keys match. We will omit F and write $\bowtie(k)$ where k is the name of the key. An example join of employee and department tables is given in Figure 5. The result of the join of two tables can itself be joined against another table allowing an unlimited number of tables to be joined together.

In `OpenMx`, joins were facilitated by a modest change to the user interface. Two parameters, `joinKey` and `joinModel`, were added to `mxMatrix` and `mxPath`, and `primaryKey` was added to `mxData`. `MxMatrix` objects are always contained in an `MxModel`. We will call this model the `MxMatrix`'s home model. When a join is performed, the specified `joinModel` is joined against the home model using the `joinKey` column in the home model to match against the `primaryKey` column in the `joinModel`. For `mxPath`, a more friendly interface was devised, naming the join model in the `from` parameter (i.e., `from='joinModel.column'`).

An alternate way to store associations in a relational database is to use a separate linking table. For example, a *classroom membership* table might contain foreign keys for both teacher and student. A linking table facilitates many-to-many relationships. A teacher can have many students and a student can have many teachers. Although there is no problem with linking tables from the standpoint of the join operator, it problematic from a modeling point of view because the maximum number of teachers per student is not fixed. How can the student model be specified? We leave this question to future research.

Mixed model, details

Although the user interface is less flexible and convenient compared to RAM, the mixed model is important because a great deal of research has gone into its efficient estimation (e.g., Bates & DebRoy, 2004; Harville, 1977; Lindstrom & Bates, 1990; Searle, Casella, & McCulloch, 1992; Wolfinger, Tobias, & Sall, 1994). Recent work has generalized the mixed model to non-Gaussian distributions (Rabe-Hesketh, Skrondal, & Pickles, 2004; Skrondal & Rabe-Hesketh, 2004), but we restrict our focus to Gaussian models. More detailed expositions of the mixed model are available from many sources (e.g., Bates, Mächler, Bolker, & Walker, 2014; West, Welch, & Galecki, 2014). The essentials are as follows.

In matrix notation, for column vector of observations $\boldsymbol{Y}$, covariates $\boldsymbol{X}$ associated with constant coefficients $\boldsymbol{\beta}$, covariates $\boldsymbol{Z}$ associated with varying coefficients $\boldsymbol{u}$, and column vector of residuals $\boldsymbol{e}$, the mixed model can be written as,

$$\boldsymbol{Y} = \underbrace{\boldsymbol{X}\boldsymbol{\beta}}_{\text{constant}} + \underbrace{\boldsymbol{Z}\boldsymbol{u} + \boldsymbol{e}}_{\text{varying}}. \tag{28}$$

We assume $\boldsymbol{u}$ and $\boldsymbol{e}$ are normally distributed with

$$\mathrm{E}\begin{pmatrix}\boldsymbol{u}\\ \boldsymbol{e}\end{pmatrix} = \begin{pmatrix}\boldsymbol{0}\\ \boldsymbol{0}\end{pmatrix} \tag{29}$$

$$\mathrm{Cov}\begin{pmatrix}\boldsymbol{u}\\ \boldsymbol{e}\end{pmatrix} = \begin{pmatrix}\boldsymbol{G} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{R}\end{pmatrix}. \tag{30}$$

The design matrix, $\boldsymbol{X}$, is not estimated. The matrix $\boldsymbol{Z}$ can be used in two distinct ways: as a design matrix for varying coefficients (not estimated) or as estimated factor loadings for latent factors (Skrondal & Rabe-Hesketh, 2004, p. 107).

Although Equation 28 builds intuition, it actually describes the distribution of $\boldsymbol{Y}$

conditional on a particular realization of $\boldsymbol{u} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{G})$. The unconditional distribution is

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{e} \tag{31}$$

which is essentially linear regression (c.f. Equation 2) where

$$\boldsymbol{e} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{Z}\boldsymbol{G}\boldsymbol{Z}^T + \boldsymbol{R}). \tag{32}$$

Univariate models typically use $\boldsymbol{R} = \sigma^2 \boldsymbol{I}$. Independent units of analysis in multivariate models typically use a block diagonal $\boldsymbol{R}$ with each block as the independent unit. Once covariance components $\boldsymbol{R}$ and $\boldsymbol{G}$ are estimated, analytic solutions are available for constant $\hat{\boldsymbol{\beta}}$ and varying $\hat{\boldsymbol{u}}$ coefficients (Henderson Jr, 1982),

$$\begin{pmatrix} \boldsymbol{X^T \hat{R}^{-1} X} & \boldsymbol{X^T \hat{R}^{-1} Z} \\ \boldsymbol{Z^T \hat{R}^{-1} X} & \boldsymbol{Z^T \hat{R}^{-1} Z + \hat{G}^{-1}} \end{pmatrix} \begin{pmatrix} \hat{\boldsymbol{\beta}} \\ \hat{\boldsymbol{u}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{X^T \hat{R}^{-1} Y} \\ \boldsymbol{Z^T \hat{R}^{-1} Y} \end{pmatrix}. \tag{33}$$

That is, varying coefficients $\boldsymbol{u}$ need not be estimated directly but can be obtained as an analytic function of the covariance. The solutions of Equation 33 can be written as,

$$\hat{\boldsymbol{\beta}} = \left(\boldsymbol{X}^T \hat{\boldsymbol{V}}^{-1} \boldsymbol{X}\right)^{-1} \boldsymbol{X}^T \hat{\boldsymbol{V}}^{-1} \boldsymbol{Y} \tag{34}$$

$$\hat{\boldsymbol{u}} = \hat{\boldsymbol{G}} \boldsymbol{Z}^T \hat{\boldsymbol{V}}^{-1} \left(\boldsymbol{Y} - \boldsymbol{X}\hat{\boldsymbol{\beta}}\right) \tag{35}$$

where

$$\boldsymbol{V} \equiv \boldsymbol{Z}\hat{\boldsymbol{G}}\boldsymbol{Z}^T + \hat{\boldsymbol{R}}. \tag{36}$$

For parameter vector $\boldsymbol{\theta}$, the -2 log-likelihood of n independent observations is,

$$-2\ell(\boldsymbol{\beta}, \boldsymbol{\theta}) = nk \log(2\pi) + \log|\boldsymbol{V}| + (\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta})^T \boldsymbol{V}^{-1} (\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta}) \tag{37}$$

where k is the size of $\boldsymbol{V}$. This likelihood can be simplified by plugging Equation 34 in for $\boldsymbol{\beta}$ (using provisional estimates). The resulting profile −2 log-likelihood is,

$$-2\ell(\boldsymbol{\theta}) = nk\log(2\pi) + \log|\boldsymbol{V}| + \boldsymbol{r}^T\boldsymbol{V}^{-1}\boldsymbol{r} \tag{38}$$

where

$$\boldsymbol{r} = \boldsymbol{Y} - \boldsymbol{X}\left[\left(\boldsymbol{X}^T\boldsymbol{V}^{-1}\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{V}^{-1}\boldsymbol{Y}\right]. \tag{39}$$

This likelihood does not take into account the loss of degrees of freedom from constant coefficients $\boldsymbol{\beta}$ in the estimation of covariance parameters $\boldsymbol{\theta}$. Uncorrected, covariance parameters tend to exhibit bias. A solution was proposed to obtain unbiased covariance parameters estimates (known as REML; Patterson & Thompson, 1971). The REML approach can be implemented in `OpenMx` (Cheung, 2013). However, when REML is used, the likelihood ratio test cannot be used for constant coefficients $\boldsymbol{\beta}$ (West et al., 2014, p. 35). Fortunately, the addition of a Wishart prior to the likelihood corrects bias even more accurately than REML (Chung, Gelman, Rabe-Hesketh, Liu, & Dorie, 2015). The addition of a Bayesian prior is an elegant solution that corrects for bias without impairing the posterior ratio test.

Inference

Large sample theory provides a number of ready tools for inference such as the Wald test (including the sandwich estimator), the likelihood ratio test (including profile likelihood confidence intervals), the bootstrap, and the jackknife (Pawitan, 2001; Pek & Wu, in press; White, 1982). Results established using the mixed model apply to corresponding relational SEM models. For example, improvement in precision is possible by conditioning on the type of inference being considered. For constant coefficients, adjustments are advised to improve calibration of the false positive rate (e.g., Manor & Zucker, 2004). Inference on variance components can be divided into two cases. When the null hypothesis does not involve a parameter space boundary then standard asymptotic results apply. An example is a test between heterogeneous and homogeneous residual variance. The second case arises when a parameter space boundary is involved. This commonly occurs in the test of whether to include a varying coefficient because varying coefficients are not tested directly but by restriction of their variance (and covariances) to zero (e.g., Crainiceanu & Ruppert, 2004).

While inference for relational SEM builds on prior research, new model structures may require new inference guidelines. Inference in multilevel models is an evolving area. More research is needed.

The mixed model in `OpenMx`

Instead of following notation similar to that in use by relational databases, a model specification syntax inspired by conditional probability notation evolved in some popular `R` packages that implement the mixed model (e.g., Bates et al., 2014; Pinheiro, Bates, DebRoy, Sarkar, & R Core Team, 2016). Formula notation (Wilkinson & Rogers, 1973) for specifying a regression equation was augmented with a vertical bar clause,

```
lmer(Reaction ~ Days + (Days | Subject), sleepstudy)
```

The left part of the regression equation, up to the parenthesis enclosing the vertical bar, follows standard formula notation. The vertical bar clause is used to specify varying coefficients. The part after the vertical bar (`Subject`) names a factor (a column in the data frame) that partitions the data set. The formula before the vertical bar (`Days`) is joined to the base model according to this factor. The implied relational model may be clarified by translation to an equivalent `OpenMx` model. While the model specification will be longer and more laborious, `OpenMx` will offer greater flexibility and permit models that are impossible with `lmer`.

```
bySubj <- mxModel(
```

```
2      model="bySubj", type="RAM",
3      latentVars=c("slope", "intercept"),
4      mxData(data.frame(Subject=unique(sleepstudy$Subject)),
5             type="raw", primaryKey = "Subject"),
6      mxPath(c("intercept", "slope"), arrows=2, values=1),
7      mxPath("intercept", "slope", arrows=2, values=.25, labels="cov1"))

8
9  ss <- mxModel(
10     model="sleep", type="RAM", bySubj,
11     manifestVars="Reaction", latentVars = "Days",
12     mxData(sleepstudy, type="raw", sort=FALSE),
13     mxPath("one", "Reaction", arrows=1, free=TRUE),
14     mxPath("one", "Days", arrows=1, free=FALSE, labels="data.Days"),
15     mxPath("Days", "Reaction", arrows=1, free=TRUE),
16     mxPath("Reaction", arrows=2, values=1),
17     mxPath(paste0('bySubj.', c('intercept','slope')),
18            'Reaction', arrows=1, free=FALSE, values=c(1,NA),
19            labels=c(NA, "data.Days"), joinKey="Subject"))
```

We create an `mxModel` to contain the per-`Subject` model (line 1). Traditionally, the mixed model does not permit manifest observations in upper levels. Hence, upper levels only contain latent variables (line 3). The `Subject` model's data contains no observations, only primary keys (line 4). Conceptually, we would like to allow a per-`Subject` coefficient for `intercept` and `slope`. It may be surprising that this is accomplished by estimating the variance of those varying coefficients and not the coefficients themselves (line 6). We estimate the covariance between varying `intercept` and `slope` (line 7).

We include the upper level model as a submodel of the base model (line 10). The rationale for this organization and other possible organizations are discussed in Figure 6. The `lme4` package offers a double vertical bar notation to indicate that the

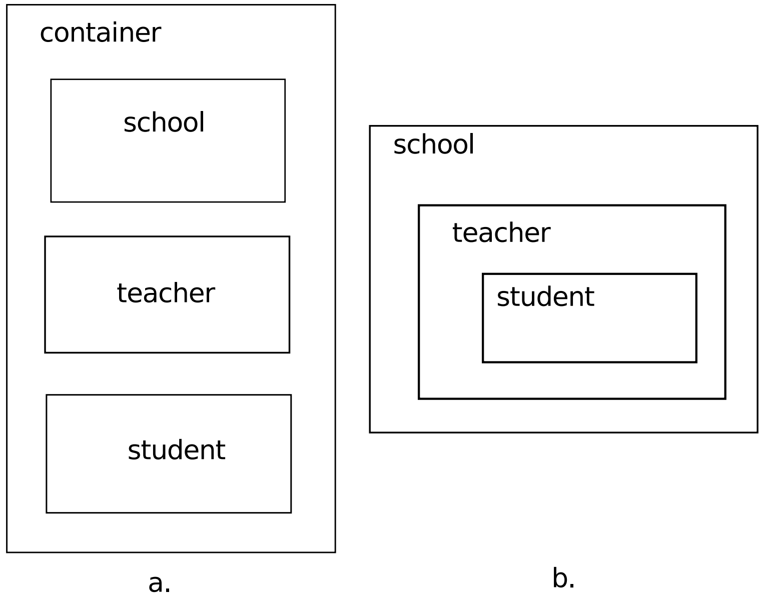

Figure 6. Two equivalent model specifications for students nested within teachers nested within schools. Each rectangle corresponds to an `mxModel`. The prototype used organization (a) to specify nested multilevel models. We finalized on (b) for `mxPath` specified models. Scheme (b) may seems backwards, but it offers the advantage that each submodel is also a valid model. This is due to the constraint that outer models cannot depend on inner models. For example, a school cannot depend on a teacher and a teacher cannot depend on a student. This structure is only required for `mxPath` specified models. No particular model nesting is required for `mxMatrix` specified models.

varying coefficient covariance should be fixed to zero. The constant coefficients are specified starting at line 13. The predictor `Days` is included in the model as a zero variance regression (line 14). This warrants a brief digression.

In structural equation modeling, it is customary to assume a normal distribution for both predictor and response variables. In contrast, regression models assume only that the residual is normally distributed. No distributional assumption is made about predictors. There are pros and cons to both approaches.

A major advantage of assuming a distribution for predictors is that missing data are less of a problem (Enders & Bandalos, 2001). However, when predictors are not missing and predictor covariance is not of substantive interest then modeling predictors can add extra parameters for little gain. For example, a script from the

OpenMx test suite, UnivariateRandomInterceptWide.R, implements a single predictor univariate random intercept model. The standard regression approach estimates 4 parameters (residual variance, intercept, constant regression coefficient, and varying intercept variance), but UnivariateRandomInterceptWide.R also estimates the mean and variance of predictor X, adding 2 parameters for a total of 6 (see Appendix A). The parameters that are common among these two models have matching estimates, but why estimate an extra 2 parameters unless they are of substantive interest? For optimal performance, the analyst should think carefully about whether a predictor needs to be modeled as normally distributed or can be included in the model as a zero variance regression.

The connections between the per-Subject and base models are set up at line 17. These connections correspond to the Z matrix in Equation 28. An executable version of this code is available in Appendix B. While the OpenMx is not as succinct as lmer, the OpenMx model could easily be extended to incorporate multivariate data such as digit span in addition to reaction time. Another lmer example using the Orthodont data set is available in Appendix C.

All mixed models can be similarly translated into OpenMx models. Each vertical bar clause is implemented with a latent mxModel of extra variance to account for the varying coefficients. These latent OpenMx models are joined to the corresponding constant coefficients in the base model using fixed loadings. Although standard practice is to estimate varying coefficients with a variance, one script in the OpenMx test suite, MultilevelUniRandomSlopeInt.R, estimates the varying coefficients themselves. A corresponding model that estimates a varying coefficient variance has been added to this script (Appendix D).

Upper to lower level transition matrices can take advantage of the usual OpenMx capabilities. A transition matrix can contain free parameters, definition variables, or populated values using square bracket notation. Or for maximum flexibility, transition matrices can be specified as the result of an `mxAlgebra`.

Speeding up nested multilevel

We will trace through in more technical detail the steps involved in optimization of nested multilevel structure. Nested varying coefficients produce a sparse covariance matrix with a pattern amenable to an efficient inverse (Goldstein & McDonald, 1988), but we will do better. We review how the Gaussian distribution is invariant to orthogonal rotation, show how to use the QR decomposition algorithm to create a rotation to specific axis vectors, and introduce the novel Rampart rotation to dramatically improve independence in multilevel covariance matrices. Rampart performance benefits and limitations are described. To validate the implementation, we finish with a simulation study.

Rampart can only be applied to nested multilevel structure. Crossed varying coefficients create less orderly covariance patterns. When Rampart is not applicable to a sub-problem, `OpenMx` uses sparse matrix algebra to compute inverses for arbitrarily crossed models (Fellner, 1987).

Topological sort

Once a relational SEM is specified, each row must be assigned to a location in a model-wide covariance matrix (Goldstein & McDonald, 1988). There are many possible assignments of rows to covariance locations. One type of ordering that offers a computational advantage is a topological sort. We can regard a relational SEM as a directed graph. If we add the restriction that cycles are not allowed then we can sort the graph by dependency. Units without dependency on other units can come first and then dependent units. For example, refer to Figure 7. This ordering allows us to compute the model expected mean unit-wise instead of model-wise.

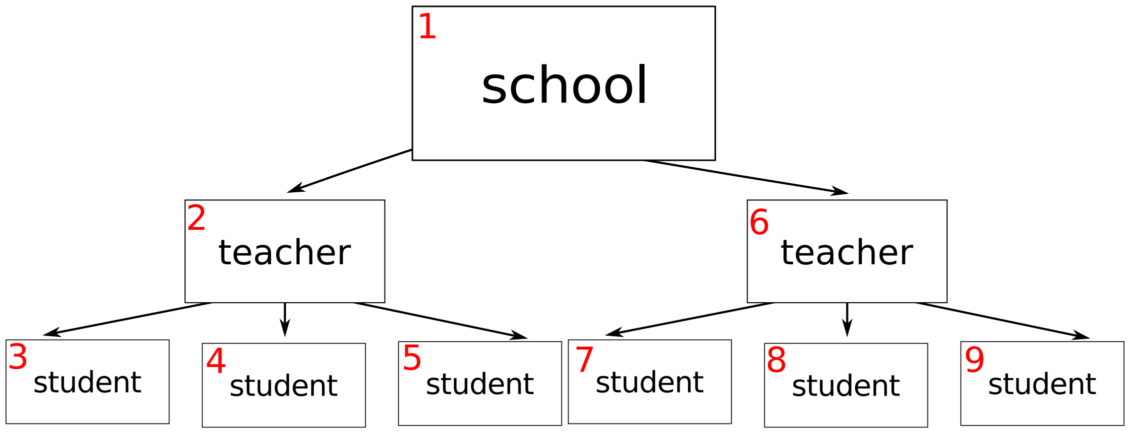

Figure 7. Topological sort is accomplished by depth-first search (Tarjan, 1976) in the opposite direction of the arrows starting from each of the lowest level units (students in this example). Units are assigned a location (the number in red) as soon as all the units that they depend upon are assigned a location. This algorithm is linear in time with the number of units.

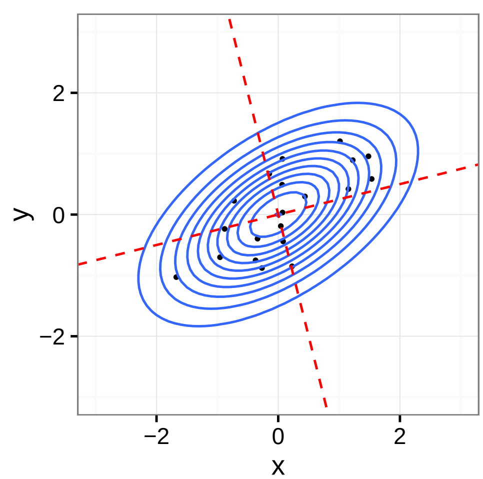

Figure 8. Observations (represented by points) in a Gaussian density. The likelihood of these points is unaffected by axis rotation. For example, the axis could be rotated to the red dashed lines without affecting the likelihood.

Gaussian density rotation

An intuitive argument is given in Figure 8. Here we work through the equations to understand exactly how an orthogonal rotation $\boldsymbol{Q}$ fits into the Gaussian likelihood. The -2 log density of a single observation x from the K dimensional Gaussian distribution is,

$$K \log(2\pi) + log(|\boldsymbol{\Sigma}|) + (\boldsymbol{\mu} - \boldsymbol{x})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{\mu} - \boldsymbol{x}). \tag{40}$$

Suppose we want to apply an orthogonal rotation $\boldsymbol{Q}$ to $\boldsymbol{x}$. The rotated $\boldsymbol{Q}$ density is,

$$K \log(2\pi) + log(|\boldsymbol{Q}\boldsymbol{\Sigma}\boldsymbol{Q}^T|) + (\boldsymbol{Q}(\boldsymbol{\mu} - \boldsymbol{x}))^T \boldsymbol{Q}\boldsymbol{\Sigma}^{-1}\boldsymbol{Q}^T \left(\boldsymbol{Q}(\boldsymbol{\mu} - \boldsymbol{x})\right). \tag{41}$$

We know that $|\boldsymbol{Q}\boldsymbol{\Sigma}\boldsymbol{Q}^T|$ is equal to $|\boldsymbol{\Sigma}|$ because $\boldsymbol{Q}$ is an orthogonal transformation and eigenvalues are preserved. For the term on the right, we can expand the transpose, regroup, and use the fact that $\boldsymbol{Q}^{-1} = \boldsymbol{Q}^T$,

$$(\boldsymbol{Q}(\boldsymbol{\mu} - \boldsymbol{x}))^T \boldsymbol{Q}\boldsymbol{\Sigma}^{-1}\boldsymbol{Q}^T \left(\boldsymbol{Q}(\boldsymbol{\mu} - \boldsymbol{x})\right) \tag{42}$$

$$\left((\boldsymbol{\mu} - \boldsymbol{x})^T \boldsymbol{Q}^T\right) \boldsymbol{Q}\boldsymbol{\Sigma}^{-1}\boldsymbol{Q}^T \left(\boldsymbol{Q}(\boldsymbol{\mu} - \boldsymbol{x})\right) \tag{43}$$

$$(\boldsymbol{\mu} - \boldsymbol{x})^T (\boldsymbol{Q}^T\boldsymbol{Q}) \boldsymbol{\Sigma}^{-1} (\boldsymbol{Q}^T\boldsymbol{Q}) (\boldsymbol{\mu} - \boldsymbol{x}) \tag{44}$$

$$(\boldsymbol{\mu} - \boldsymbol{x})^T I \boldsymbol{\Sigma}^{-1} I (\boldsymbol{\mu} - \boldsymbol{x}) \tag{45}$$

$$(\boldsymbol{\mu} - \boldsymbol{x})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{\mu} - \boldsymbol{x}). \tag{46}$$

QR decomposition

QR decomposition is a versatile procedure that can be used to accomplish a variety of goals. QR decomposition expresses matrix $\boldsymbol{A}$ as the product of orthogonal matrix $\boldsymbol{Q}$ and upper triangular matrix $\boldsymbol{R}$. Matrix $\boldsymbol{A}$ must be m-by-n with $m \geq n$. Here we describe how to use the QR decomposition algorithm to create an orthogonal axis rotation that we can plug into the Gaussian density (Equation 41). Hence, $\boldsymbol{A}$ will always be m-by-m (square) and full rank. Let $\boldsymbol{x}$ be an arbitrary column vector of $\boldsymbol{A}$ of length $|\alpha|$. One Householder reflection consists of,

$$\boldsymbol{u} = \boldsymbol{x} + \operatorname{sign}(x_1)\alpha\,[1, 0, \ldots, 0]^T \tag{47}$$

$$\boldsymbol{v} = \frac{\boldsymbol{u}}{||\boldsymbol{u}||} \tag{48}$$

$$\boldsymbol{Q} = \boldsymbol{I} - 2\boldsymbol{v}\boldsymbol{v}^T. \tag{49}$$

In Equation 47, we choose the sign to increase the magnitude of the first entry of $\boldsymbol{x}$. This ensures the length of $\boldsymbol{u}$ is at least α. Vector $\boldsymbol{u}$ can be regarded as the average of the direction of $\boldsymbol{x}$ and the target axis. Vector $\boldsymbol{v}$ is the reflection pivot. The obtained $\boldsymbol{Q}$ will zero out all except the first row of $\boldsymbol{x}$ such that,

$$\boldsymbol{QA} = \begin{bmatrix} \alpha_1 & \star & \ldots & \star \\ 0 & & & \\ \vdots & & \boldsymbol{A}' & \\ 0 & & & \end{bmatrix}. \tag{50}$$

The process is repeated on $\boldsymbol{A}'$ until $\boldsymbol{QA}$ is upper triangular, generating a series of rotations $\boldsymbol{Q_1}$, $\boldsymbol{Q_2}$, $\ldots$, $\boldsymbol{Q_m}$.

To illustrate the process, let us perform a rotation to an arbitrary basis,

$$\boldsymbol{A} = \begin{bmatrix} 2.87 & & \\ 2.55 & 2.88 & \\ 1.27 & 2.88 & 0.91 \end{bmatrix}. \tag{51}$$

We place the basis vectors in the lower triangle because the QR algorithm is blind to the upper triangle. The first reflection obtains,

$$\boldsymbol{x_1} = \begin{bmatrix} 2.87 \\ 2.55 \\ 1.27 \end{bmatrix} \tag{52}$$

$$\alpha_1 = ||\boldsymbol{x_1}|| = 4.04 \tag{53}$$

$$\boldsymbol{u} = \boldsymbol{x_1} + \text{sign}(x_{1,1})\alpha_1 \left[1, 0, \ldots, 0\right]^T = \begin{bmatrix} 6.91 \\ 2.55 \\ 1.27 \end{bmatrix} \tag{54}$$

$$\boldsymbol{v} = \frac{\boldsymbol{u}}{||\boldsymbol{u}||} = \begin{bmatrix} 0.92 \\ 0.34 \\ 0.17 \end{bmatrix} \tag{55}$$

$$\boldsymbol{Q_1} = \boldsymbol{I} - 2\boldsymbol{v}\boldsymbol{v}^T = \begin{bmatrix} \text{-0.71} & \text{-0.63} & \text{-0.31} \\ \text{-0.63} & 0.77 & \text{-0.12} \\ \text{-0.31} & \text{-0.12} & 0.94 \end{bmatrix}. \tag{56}$$

As expected, $\boldsymbol{Q_1}$ zeros all but the first entry of the first column of $\boldsymbol{A}$,

$$\boldsymbol{Q_1}\boldsymbol{A} = \begin{bmatrix} \text{-4.04} & \text{-2.72} & \text{-0.29} \\ & 1.88 & \text{-0.11} \\ & 2.38 & 0.86 \end{bmatrix}.$$

We continue with the second reflection,

$$\boldsymbol{x_2} = \begin{bmatrix} 1.88 \\ 2.38 \end{bmatrix} \tag{57}$$

$$\alpha_2 = ||\boldsymbol{x_2}|| = 3.04 \tag{58}$$

$$\boldsymbol{u} = \boldsymbol{x_2} + \text{sign}(x_{2,1})\alpha_2 \left[1, 0, \ldots, 0\right]^T = \begin{bmatrix} 4.92 \\ 2.38 \end{bmatrix} \tag{59}$$

$$\boldsymbol{v} = \frac{\boldsymbol{u}}{||\boldsymbol{u}||} = \begin{bmatrix} 0.90 \\ 0.44 \end{bmatrix} \tag{60}$$

$$\boldsymbol{Q_2} = \boldsymbol{I} - 2\boldsymbol{v}\boldsymbol{v}^T = \begin{bmatrix} 1.00 & & \\ & \text{-0.62} & \text{-0.79} \\ & \text{-0.79} & 0.62 \end{bmatrix}. \tag{61}$$

$\boldsymbol{Q_2}$ is 2-by-2, but we fill it with the identity matrix to expand it back to m-by-m. $\boldsymbol{A}$ is fully decomposed. We obtain,

$$\boldsymbol{Q} = \boldsymbol{Q_2}\boldsymbol{Q_1} = \begin{bmatrix} \text{-0.71} & \text{-0.63} & \text{-0.31} \\ 0.64 & \text{-0.38} & \text{-0.67} \\ 0.30 & \text{-0.67} & 0.67 \end{bmatrix} \tag{62}$$

$$\boldsymbol{R} = \boldsymbol{Q_2}\boldsymbol{Q_1}\boldsymbol{A} = \begin{bmatrix} \text{-4.04} & \text{-2.72} & \text{-0.29} \\ & \text{-3.04} & \text{-0.61} \\ & & 0.62 \end{bmatrix} \tag{63}$$

However, this $\boldsymbol{Q}$ is the inverse of what we want. We want the rotation from the identity axis to the axis described by $\boldsymbol{A}$. Hence, the desired rotation is $\boldsymbol{Q}^T$. With a deeper understanding of axis rotation, we have the tools we need to describe the Rampart rotation.

Rampart rotation

Let us take a close look at the model in Figure 9. This model is identified with only two teachers. With only 8 observations, the matrices are compact enough to investigate the full model. First we examine the model implied covariance (Equation 20). Our model has no latent variables so the $\boldsymbol{F}$ matrix is set to the identity.

Parameters are assigned arbitrary values.

$$\boldsymbol{A} = \begin{bmatrix} & & & \\ 1.07 & & & \\ 1.07 & & & \\ 1.07 & & & \end{bmatrix} \tag{64}$$

$$\boldsymbol{S} = \begin{bmatrix} 0.29 & & & \\ & 0.70 & & \\ & & 0.70 & \\ & & & 0.70 \end{bmatrix} \tag{65}$$

$$\boldsymbol{\Sigma} = (\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{S}(\boldsymbol{I} - \boldsymbol{A})^{-T} = \begin{bmatrix} 0.29 & & & \\ 1.06 & 0.70 & & \\ 1.06 & & 0.70 & \\ 1.06 & & & 0.70 \end{bmatrix} \tag{66}$$

We obtain a 4-by-4 covariance matrix instead of 8-by-8 since both sets of teacher-and-students have the same model. However, this efficiency gain of grouping by independence does not help much if we add more students. A classroom with a few hundred students is going to require a large covariance matrix.

Observe that λ, the regression from teacher to student, is a single parameter that is some function of the mean of the students. This is true regardless of the number of students. Instead of distributing the information about the mean across all the students, suppose we could rotate the data such that the mean was already computed and readily available. In fact, we can.

Let us use a QR decomposition find an orthogonal rotation to basis vectors,

$$\begin{bmatrix} 1.00 & 2.00 & \\ 1.00 & \text{-}1.00 & 1.00 \\ 1.00 & \text{-}1.00 & \text{-}1.00 \end{bmatrix}. \tag{67}$$

These vectors are not normalized to unit length to make it easier to understand the construction. The first column vector obtains a value proportional to the mean. The remaining basis vectors consist of an arbitrary orthogonal contrast, Helmert contrasts in this case. QR decomposition obtains

$$\boldsymbol{Q}^T = \begin{bmatrix} \text{-}0.58 & \text{-}0.58 & \text{-}0.58 \\ 0.82 & \text{-}0.41 & \text{-}0.41 \\ & \text{-}0.71 & 0.71 \end{bmatrix}. \tag{68}$$

We apply this rotation to the 3 student values associated with the first teacher,

$$\boldsymbol{Q}^T \begin{bmatrix} 0.69 \\ \text{-}2.03 \\ \text{-}0.98 \end{bmatrix} = \begin{bmatrix} 1.34 \\ 1.79 \\ 0.74 \end{bmatrix}. \tag{69}$$

The mean of the first 3 students is -0.77. The value obtained (1.34) is $-\sqrt{3}$ times the mean. The wrong sign is due to rotational indeterminacy. We can take $-\boldsymbol{Q}^T$ instead of $\boldsymbol{Q}^T$. The $\sqrt{3}$ factor results from the need to preserve the length of the original vector, $\sqrt{3} = \sqrt{1^2 + 1^2 + 1^2}$. The remaining values reflect the variance,

$$\frac{\begin{bmatrix} 1.79 & 0.74 \end{bmatrix} \begin{bmatrix} 1.79 \\ 0.74 \end{bmatrix}}{3 - 1} = \text{Var} \begin{bmatrix} 0.69 \\ \text{-}2.03 \\ \text{-}0.98 \end{bmatrix} = 1.88. \tag{70}$$

With the data rotated, a corresponding rotation to the covariance matrix is required to leave the density function unchanged. We rotate the teacher-to-student regression weights. Note that the value of these weights are constant for all students, in other words, the weights have zero variance. Therefore, all of the links to the students, besides the first, get zeroed and the first link is multiplied by $\sqrt{3}$ (see Figure 10). Since $\boldsymbol{S}$ remains as in Equation 65 and the rotated asymmetric matrix

$$\boldsymbol{A}^* = \begin{bmatrix} & & & \\ 1.85 & & & \\ & & & \\ & & & \end{bmatrix}, \tag{71}$$

$$\boldsymbol{\Sigma} = (\boldsymbol{I} - \boldsymbol{A}^*)^{-1}\boldsymbol{S}(\boldsymbol{I} - \boldsymbol{A}^*)^{-T} = \begin{bmatrix} 0.29 & 0.54 & & \\ 0.54 & 1.71 & & \\ & & 0.70 & \\ & & & 0.70 \end{bmatrix}. \tag{72}$$

This rotation dramatically increases the independence in the model implied distribution. Regardless of the number of students, interdependent blocks of the covariance matrix need never be larger than 2-by-2 (and most of them are 1-by-1). Moreover, this algorithm can be applied recursively in more complex models with many levels such that most of the nonzero regions in a very large multilevel covariance structure (e.g., Equation 27) become independent. Note that the rotated $\boldsymbol{A}^*$ matrix (Equation 71) is only used to compute the covariance (Equation 20). Although $\boldsymbol{A}$ also appears in the computation of the expected means (Equation 19), this equation uses the unrotated $\boldsymbol{A}$. The residuals are rotated, not (somehow) the predicted means (refer to Equation 41).

To extend this univariate approach to multiple indicators per students, we can rotate each indicator independently. Since the orthogonal contrasts are identical and

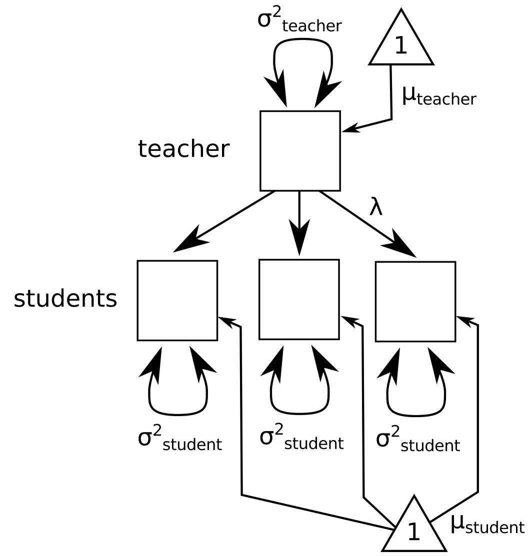

Figure 9. A simple multilevel model with 5 parameters: $\sigma^2_{teacher}$, $\mu_{teacher}$, $\sigma^2_{student}$, $\mu_{student}$, and λ. The three students have exactly the same model implied distribution.

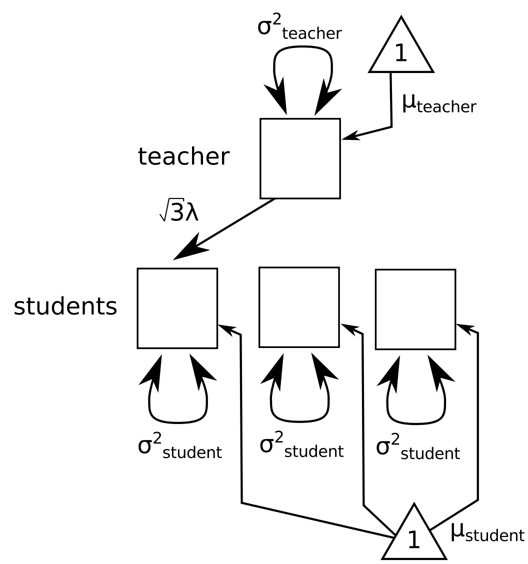

Figure 10. Figure 9 after Rampart rotation is applied to unlink all but one student from the teacher. Note that the student data (not shown) requires a corresponding rotation to preserve the value of the likelihood.

in the same order for each indicator, not only is the variance preserved but also the covariance! Hence, there is no limit on the complexity of the student model. The only requirement is that all student models must be identical and have the same single parent.

Rampart: History and name

The idea for Rampart developed out of discussions among Timo von Oertzen, Steven M. Boker, and Timothy R. Brick during the summer of 2012. During spring 2013, Rampart was prototyped in `OpenMx` (see merge v2.3.1-294-g9968ddc in the source code repository). The prototype was limited to the situation where there are exactly the same number of lower level units for each upper level unit and no missing data. Such perfectly balanced data are unlikely to occur in practice. Moreover, the prototype did not allow definition variables. Definition variables are an important `OpenMx` feature that users expect to be implemented consistently throughout `OpenMx`. These deficiencies were remedied in the present implementation. The original proof-of-concept test script was brought up-to-date with the current syntax (Appendix E).

A rotation that was a conceptual precursor to Rampart was named *pre-processed maximum likelihood* in the title of von Oertzen and Hackett (submitted). However, the phrase *pre-processed* is remarkably non-specific. Furthermore, there is nothing about the algorithm that requires maximum likelihood as a fit function as opposed to, say, unweighted least squares. Hence, none of the elements of the original name provide helpful semantic cues. We propose *Rampart.* The name *rampart* lexically emphasizes the connection with the RAM parameterization. Colloquially, a rampart is a wall built for defense. The Rampart algorithm partitions, or places a wall between, repeated identical elements to defend against poor performance.

Sufficient statistic formula for the Gaussian density

A challenge with evaluation of the Gaussian density (Equation 1) is that the covariance dimension is very large, the total number of observations in the model. Inversion of the covariance is a computationally expensive operation, roughly $O(N^3)$. One common way to speed up evaluation of the Gaussian likelihood function is to use the sufficient statistic formula. Suppose we have data of N independent observations of K-variate units. Let $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ be the model expected mean vector and covariance matrix, respectively. Let $\boldsymbol{m}$ and $\boldsymbol{S}$ be the mean vector and covariance matrix of the data, respectively. The sufficient statistic formula is,

$$-2\log L(\text{data}|\boldsymbol{\theta}) = NK\log(2\pi) + N\log(|\boldsymbol{\Sigma}|) + (N-1)\text{tr}(\boldsymbol{\Sigma}^{-1}\boldsymbol{S}) + N(\boldsymbol{\mu}-\boldsymbol{m})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}-\boldsymbol{m})). \quad (73)$$

The derivation of this formula is given in many textbooks and omitted here. The advantage of this formula is that the maximum dimension of the covariance matrix is K regardless of the number of units N. However, this formula is only applicable when the units are independent and identical. Fortunately, Rampart dramatically improves the prospects for application of the sufficient statistic formula.

Rampart and definition variables

To apply Rampart, the upper to lower level transition matrix must be exactly the same for all lower level units. Constant transition matrices, possibly with free parameters, pose no difficulty. However, no attempt is made to check whether this condition holds when the transition matrix is an `mxAlgebra` or contains square bracket populated values. If definition variables appear in the transition matrix then an attempt is made to group them by value. For example, a univariate twin model can be specified such that the upper to lower level link is either 1 or $\sqrt{0.5}$ (Appendix F).

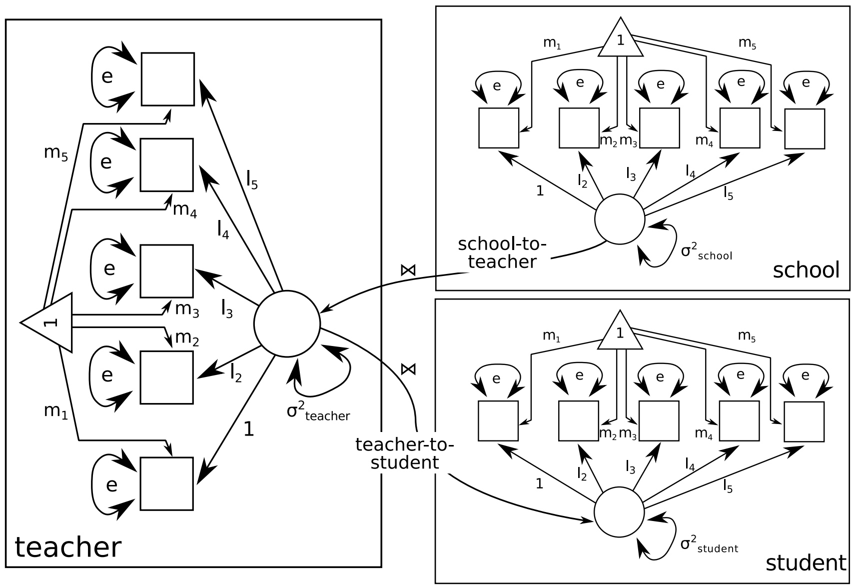

Figure 11. A 3-level latent regression model. All levels use an identical 5 indicator factor model with the loading to the first indicator fixed to 1.0, freely estimated means, free factor variance, and homogeneous error variance. Regressions are estimated from school to teacher and from teacher to school. There are 11 parameters per level and 2 between level regressions for a total of 35 parameters. Indicator error variance does not need to be homogeneous. More complex error structures are possible, but were not included in this study. Manifest indicators are not shared by levels, but are unique to their level. For example, teacher indicators might include *level of education* and *years of service*.

Rampart automatically groups same values together and transforms as many units as possible. Another common use for definition variables is to specify zero variance regressions. Since these regressions do not affect the covariance, units that differ only in mean structure are Rampart rotated and evaluated using the sufficient statistic formula (Equation 73). A model that greatly benefits from automatic identification of zero variance regressions is given in Appendix G.

Latent regression parameter recovery simulation study

To validate the accuracy of Rampart, a parameter recovery simulation study was conducted on a 3-level latent regression model. Figure 11 exhibits the per-level model structure. In addition, the first student indicator was set to missing with 20% prob-

Table 4

Euclidean norm of Monte Carlo bias and variance of parameter estimates by algorithm and parameter set. Rampart exhibits slightly less bias and variance on $\boldsymbol{\theta}_1$. Both algorithms exhibit roughly equal performance on $\boldsymbol{\theta}_2$.

$\boldsymbol{\theta}$	replications	method	$\|\|$bias$\|\|$	$\|\|\sigma^2\|\|$
1	174	rampart	1.686	0.769
		regular	1.702	0.780
2	171	rampart	2.336	0.557
		regular	2.335	0.560

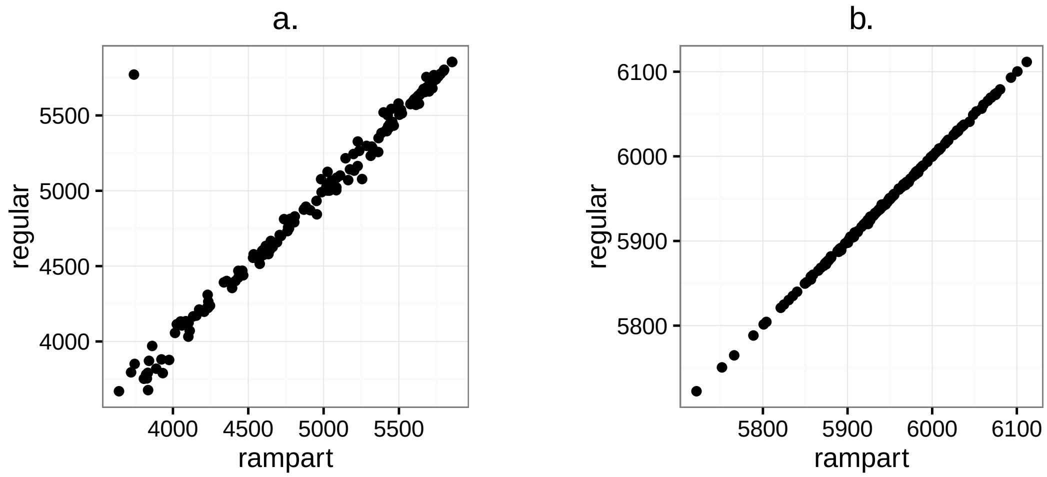

Figure 12. Scatterplot of deviance at the maximum likelihood for $\boldsymbol{\theta}_1$ (a) and $\boldsymbol{\theta}_2$ (b). In replications of $\boldsymbol{\theta}_1$, it was not uncommon for the deviance difference to be greater than 10 points. For one replication of $\boldsymbol{\theta}_1$, the regular algorithm got stuck in a local minimum more than 1000 deviance points from a better minimum found by Rampart.

ability. With observations at multiple levels, this model was outside the capability of freely available mixed model software and would be challenging to specify in SEM software without a relational join operator. The simulation study focused on validation of Rampart, comparing Rampart with the standard, unoptimized approach (i.e., simple application of Equation 1).

Two sets of true parameters ($\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$) were randomly chosen and data generated. Random numbers of students were assigned to each class and random numbers of teachers assigned per school. Parameter $\boldsymbol{\theta}_1$ was paired with 7 schools, 38 teachers, and 293 students. Parameter $\boldsymbol{\theta}_2$ was paired with 7 schools, 37 teachers, and 296 students. This was the smallest 3-level data set that we found empirically identified

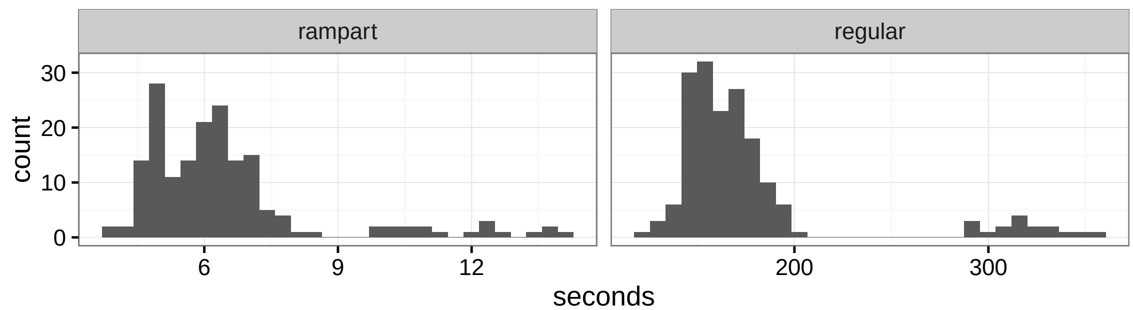

Figure 13. Seconds required per replication by algorithm for $\boldsymbol{\theta}_1$. As expected, Rampart exhibits a huge efficiency advantage on this type of model. Note the difference in scale on the x axis. Timing data for $\boldsymbol{\theta}_2$ is similar, and therefore, is omitted.

for most replications. A 4-level model (adding *district* as a higher level) was prepared to further validate the Rampart implementation (see Appendix H), but evaluation of this model using the standard algorithm required so much CPU time that a simulation study was deemed impractical.

Two hundred Monte Carlo replications were run for each condition (Algorithm$\times\theta$). For each replication, data were generated from the true parameters. The number of units, which lower level units were linked to which upper level units, and data missingness patterns were identical for all replications. The model was optimized against these data to obtain $\hat{\boldsymbol{\theta}}$, using the true parameters as starting values. For R replications, Monte Carlo bias and variance are

$$MC_{bias} \equiv \left[R^{-1}\sum_{r=1}^{R}\hat{\boldsymbol{\theta}}_r\right] - \boldsymbol{\theta}_{true} \tag{74}$$

$$MC_{var} \equiv \text{Var}(\hat{\boldsymbol{\theta}}). \tag{75}$$

After every replication, the information matrix was estimated by 2-iteration Richardson extrapolation of the central difference. The condition number of the information matrix is the maximum singular value divided by the minimum singular value and provides a rough gauge of the stability of a solution (Luenberger & Ye, 2008, p. 239). Replications were excluded from further analysis when the condition number of the information matrix was larger than 5 median absolute deviations from the median.

Results are summarized in Table 4. Rampart performed no worse than the standard algorithm. Additional insight into the performance of Rampart can be gleaned by plotting the fit values at the mode of the likelihood against each other (Figure 12). The mode found by Rampart can match the standard algorithm closely or differ by a considerable amount depending on the model. Another way to examine model stability is to take the difference between regular and Rampart condition numbers for the included replications. These means were 46.6 ($SE = 46.53$) and 5.24 ($SE = 1.84$) for $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$, respectively. That the means were positive suggest that the Rampart rotation may improve model stability. As expected, Rampart exhibited a huge efficiency advantage (Figure 13), mean time regular = 176.97s, mean time Rampart = 6.5s, Rampart/regular ratio = 0.04. Complete source code for the simulation study is included in Appendix I.

Application

In order to demonstrate the efficacy of the Rampart algorithm, we reanalyzed data from a facial expression tracking experiment (Boker et al., 2009). When two people engage in conversation, prior research indicates that the style of their head movements tend to become more similar. In this experiment, confederates engaged in conversation with naïve participants over a video conferencing system. However, naïve participants ($n = 27$) did not see the unfiltered confederates ($n = 6$) but a computer generated avatar. To produce a convincing portrayal, confederates' facial expressions were meticulously tracked in real-time. The portrayals were sufficiently convincing that no naïve participants guessed that the computer generated faces were not unmodified live video.

In a crossed experimental design, damping was applied to confederate facial expressions, vocal inflections, and head movements. Confederates were familiar with the nature of the manipulations and their probable effects, but were blind to order and timing. The head movements of both participants in the conversation were motion tracked at 81.6 Hz. The dependent variables were anterior-posterior (A-P) and lateral head angle. These correspond to nods of affirmation (pitch) and head shakes of disagreement (yaw), respectively. Vigor of angular velocity was taken as a metric. Based on prior research, it was hypothesized that women would nod and shake their head with greater vigor than men. In addition, it was hypothesized that each of the manipulations would increase the vigor of nods and shakes. The notion of vigor was operationalized as the root mean square (RMS) of the angular velocity during a condition.

For each 1 minute condition, there were 4860 velocity measurements ($81.6 \cdot 60 \approx 4860$). Conversations were described as lasting 8 minutes (Boker et al., 2009, p. 3488) with a different condition every minute. However, conversations ranged from 6 to 10 minutes with a median of 9 minutes. Conditions always lasted 1 whole minute so conversations shorter than 8 minutes did not include all conditions and conversations longer than 8 minutes included some repeated conditions.

Table 5
Comparison between a variety of modeling options. Model original *fits both anterior-posterior and lateral RMS angular velocity in a single model but leaves them independent (as a multiple group model). This matches the original model from Boker et al. (2009). Model* only_confed *adds a varying intercept for confederates. Model* xyCov *is the same as Model* original *but adds a covariance between anterior-posterior and lateral RMS angular velocity. Model* xyCov_confed *adds a varying intercept for confederates, and a covariance between anterior-posterior and lateral RMS angular velocity. Model* full *is similar to Model* xyCov_confed *but allows covariance between varying intercepts. See Appendix J for source code.*

base	comparison	ep	minus2LL	df	AIC	diffLL	diffdf	p
full		33	2275.2	1603	−930.8			
full	xyCov_confed	31	2275.7	1605	−934.3	0.5	2	0.79
full	xyCov	29	2329.1	1607	−884.9	53.9	4	0.00
full	only_confed	30	2373.0	1606	−839.0	97.8	3	0.00
full	original	28	2415.4	1608	−800.6	140.2	5	0.00

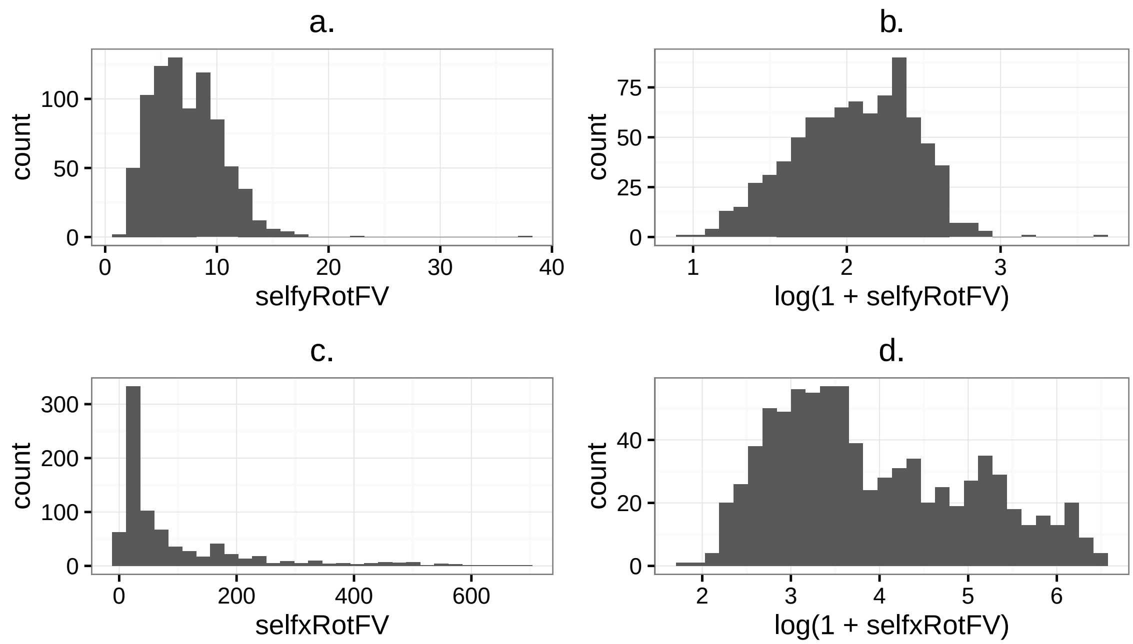

Figure 14. Anterior-posterior (a) and lateral (c) RMS angular velocity $\log(1 + x)$ transformed to (b) and (d), respectively.

The models used in the original analysis were loosely based on the Actor-Partner Interdependence Model (Cook & Kenny, 2005). These models included a varying intercept per naïve participant, but all confederates were assumed to produce equally vigorous head movements. Hence, the original model violated the assumption of independent observations since minutes involving the same confederate should be more similar than minutes involving different confederates. Another weakness in the analysis was the assumption that anterior-posterior (A-P) and lateral head angle were independent. No author believed that these two axes of head motion were independent, but no software was available to conveniently specify a multivariate model (S. Boker, personal communication, March 2015).

Before proceeding, we note that the RMS statistics are skewed and leptokurtic. The distribution can be improved by a $\log(1 + x)$ transformation (Figure 14). These raw data were carefully documented and published (Pritikin, 2016). A variety of modeling options were explored (Table 5). We selected Model *xyCov_confed* to compare against the original model.

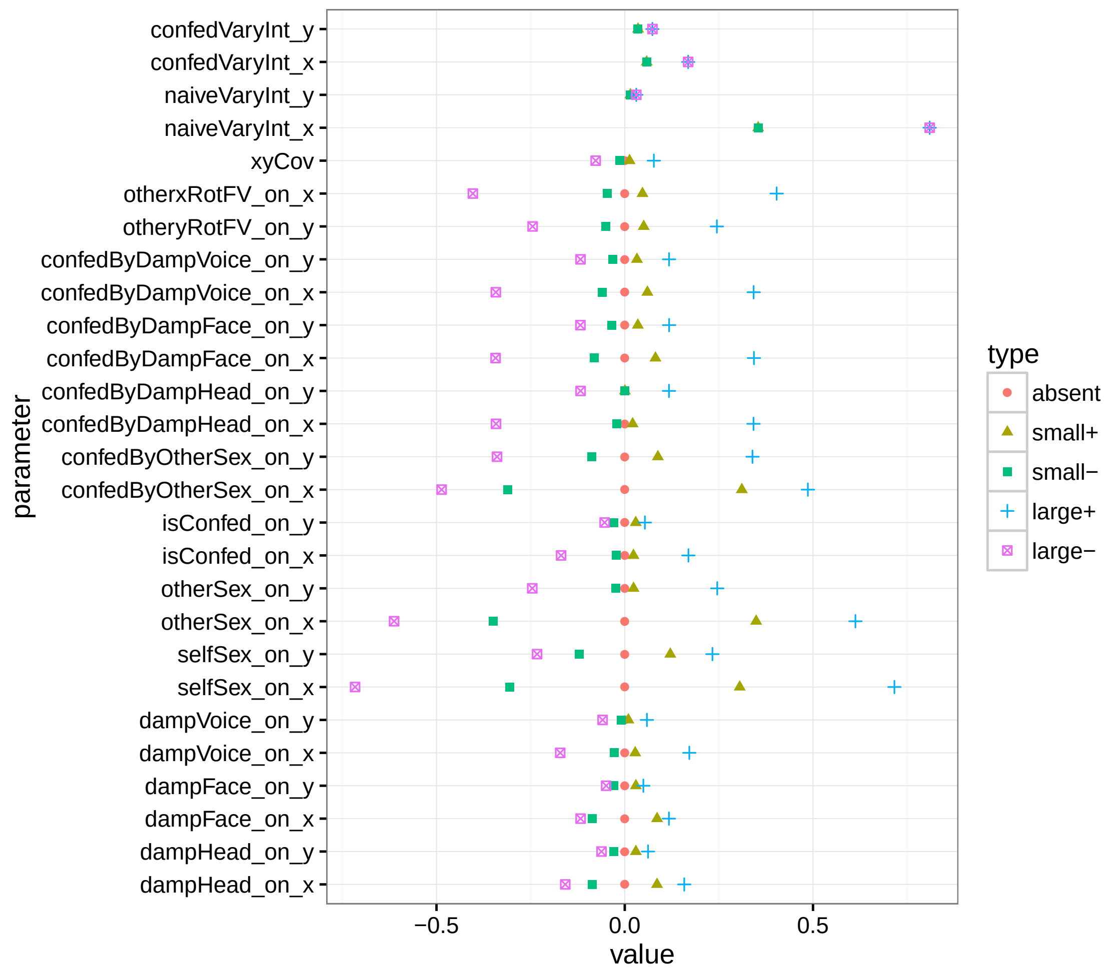

Figure 15. Generating parameters for power simulation study. For each replication, data were generated using Model *xyCov_confed* with each parameter randomly selected (with a uniform distribution) from *absent*, *small+*, *small−*, *large+*, or *large−*. Parameter values were set to correspond in magnitude with empirical parameter estimates found with Model *xyCov_confed*. An empirical parameter estimate was used in two different ways. If the parameter value divided by the standard error was 2.0 or less then it was assigned to *small* and *large* was set to 3 times the standard error. Otherwise, the parameter value was assigned to *large* and *small* was set to 1.5 the standard error. Variance parameters only used positive values. A few parameters were not of interest and used the same data generating value for all replications: the constant variances of x (lateral) and y (anterior-posterior) and their constant intercepts.

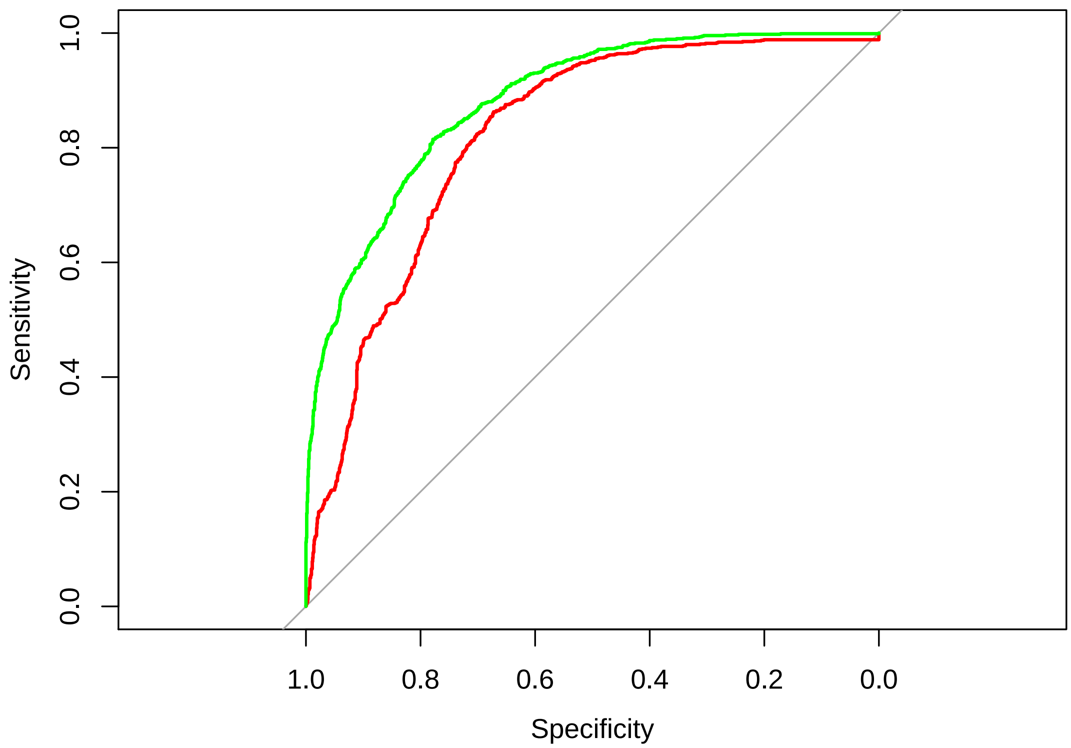

Figure 16. ROC plots for original (82.05% area under curve) and Model *xyCov_confed* (88.09% area under curve). DeLong's test of the null hypothesis that the area under the curves are equal is rejected, $D = -5.55$, $df = 4544.23$, p-$value = 3.05 \times 10^{-8}$.

A simulation study was conducted to determine how much power we might gain from Model *xyCov_confed*. Data were generated according to the scheme detailed in Figure 15. Both models were fit on 100 replications. For the original model, all replications converged but only 88 converged for Model *xyCov_confed*. Replications that failed to converge were excluded from the analysis. For each replication, the absolute parameter value divided by its standard error was taken as the quantity of evidence and the true effect was whether the corresponding generating parameter was *large*. An incorrect sign, which appeared for 12 parameter estimates throughout the simulation, was scored by negating the evidence quantity. Simulation results are summarized in Figure 16. Model *xyCov_confed* demonstrated significantly greater power on these data than the original model. Some confidence was gained that Model *xyCov_confed* can accurately recover parameters from simulated data. See

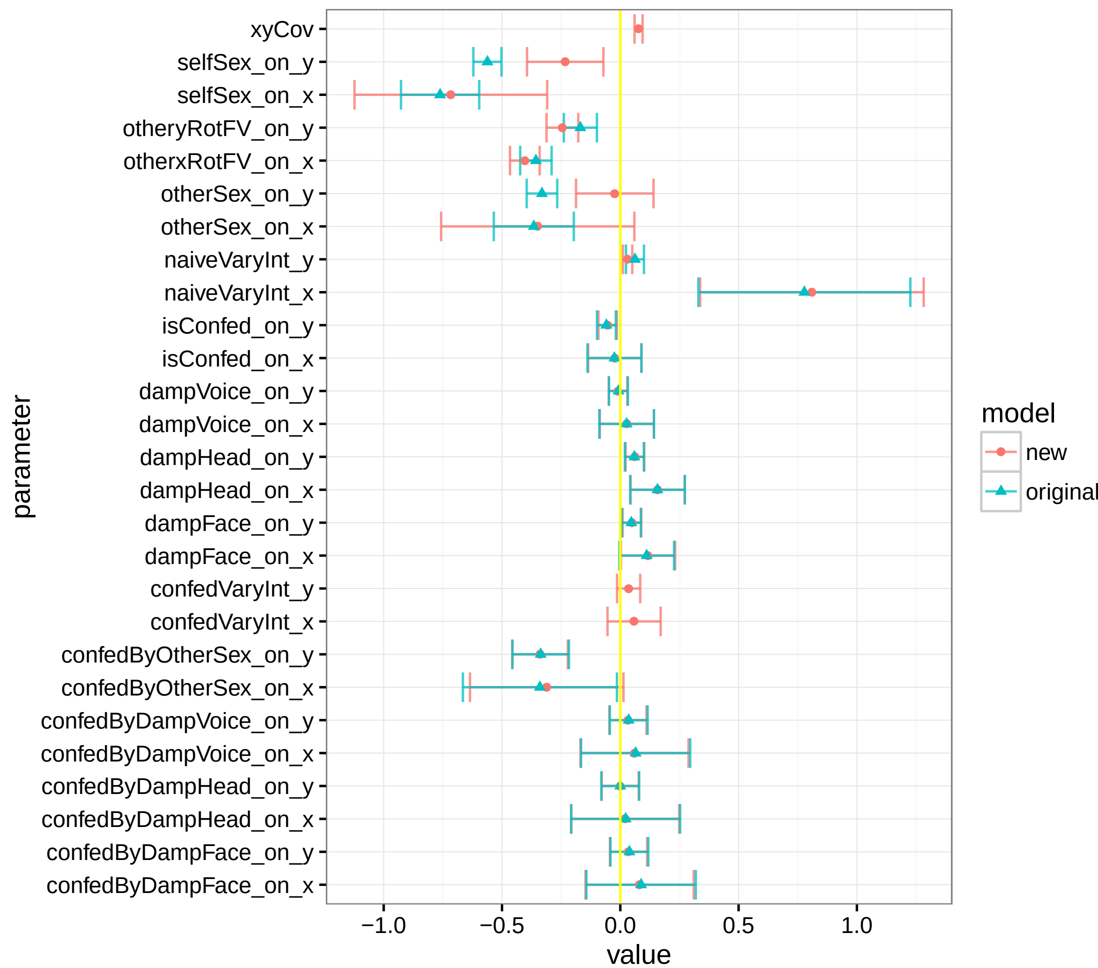

Figure 17. Parameter estimates for original and new model. Error bars represent $\pm 2SE$. Parameter *otherSex* became non-significant and the effect size of *selfSex* declined. Otherwise, most parameter estimates seemed to change little.

Appendix K for the simulation source code.

Figure 17 exhibits the original parameter estimates together with estimates from the new model. Some doubt is cast on the effect of sex on RMS angular velocity, but otherwise, most of the estimates remained stable. Although our contribution is a step forward, much more could be done to analyze these data in greater depth. For example, it is now feasible to decompose the one minute conditions into 2s chunks and estimate both within and between condition contributions. This would be computationally difficult without Rampart.

Discussion

We reviewed the development of Gaussian modeling from its beginnings in intuitive theories of causation to relational structural equation modeling. The optimization of nested multilevel models pose particular computational challenges. Rampart, a novel approach that simplifies nested multilevel structure, was devised and implemented in `OpenMx`. This implementation is of the quality required by applied researchers. A latent regression parameter recovery simulation study was conducted to demonstrate the correctness of the implementation. The implementation allows for unbalanced and missing data, and definition variables. To highlight the flexibility of the new relational SEM interface, popular mixed model regression specifications were re-expressed in `OpenMx`.

To further demonstrate Rampart, a reanalysis of Boker et al. (2009) was conducted using a multivariate model to more closely match the theoretical data generating process. In a simulation study, the multivariate model exhibited significantly higher statistical power than the original mixed model. In a comparison of the estimates obtained, most parameters did not change to a large extent except for a weaker effect of sex on head movement vigor. While the new model was an improvement on the 2009 model, the data are still highly summarized and could be modeled in greater detail given the computational efficiency of Rampart.

The join operator in `OpenMx` supports one-to-many relationships but omits support for unlimited many-to-many relationships such as can be recorded in a relational database using a linking table. For example, with a linking table, a teacher can have many students and a student can have many teachers. There is no problem with linking tables from the standpoint of the join operator, but it is not clear how to specify models that can adapt to the combination of two arbitrary sets of units.

Rampart provides a huge boost in performance, but opportunities still remain to improve performance further. For example, it is not yet clear how best to parallelize evaluation of the likelihood. The dimension of the covariance of independent groups can be large or small. The number of observations per identical covariance can be large or just a single mean vector. Further research is needed to determine the thresholds when the benefit of parallel computation outweighs the overhead of coordinating multiple threads.

Relational SEM models do not take into account the loss of degrees of freedom from constant coefficients (Patterson & Thompson, 1971). Most research to date on addressing this bias has focused on the mixed model where there is a clear delineation between constant and varying coefficients. Due to the efficiency of Rampart, it is now feasible to create relational SEM models that are nested many levels deep with some observations at each level. It is not clear whether the distinction between constant and varying coefficients applies in the circumstance where a middle level coefficient is somewhat varying and somewhat constant. The use of a Wishart prior to correct bias seems like a promising line of investigation (Chung et al., 2015). More research is needed to establish whether this approach can be profitably applied to relational SEM or whether a different approach is more suitable.

While large sample inference can rely on the asymptotic results of large sample theory, much prior research on small sample inference is limited to the mixed model (univariate with no latent factors). It is unclear whether prior research on small sample inference generalizes to relational SEM. More simulation studies are needed to provide guidance about how perform inference with small samples.

`OpenMx`, a freely available open-source statistical software package, is now capable of estimating multilevel relational structural equation models using the Rampart optimization. SEM models of large data sets, such as entire school districts, had been considered intractable due to the required estimation time. With Rampart, these data sets may now be revisited and estimated with relative efficiency.

References

Bates, D. & DebRoy, S. (2004). Linear mixed models and penalized least squares. *Journal of Multivariate Analysis*, *91*(1), 1–17.

Bates, D., Mächler, M., Bolker, B. M., & Walker, S. (2014). lme4: Linear mixed-effects models using Eigen and S4. ArXiv e-print; submitted to *Journal of Statistical Software*. Retrieved from http://arxiv.org/abs/1406.5823

Boker, S. M., Cohn, J. F., Theobald, B.-J., Matthews, I., Brick, T. R., & Spies, J. R. (2009). Effects of damping head movement and facial expression in dyadic conversation using real–time facial expression tracking and synthesized avatars. *Philosophical Transactions of the Royal Society B: Biological Sciences*, *364*(1535), 3485–3495.

Cheung, M. W.-L. (2013). Implementing restricted maximum likelihood estimation in structural equation models. *Structural Equation Modeling*, *20*(1), 157–167.

Chung, Y., Gelman, A., Rabe-Hesketh, S., Liu, J., & Dorie, V. (2015). Weakly informative prior for point estimation of covariance matrices in hierarchical models. *Journal of Educational and Behavioral Statistics*, *40*(2), 136–157. doi:10.3102/1076998615570945

Cook, W. L. & Kenny, D. A. (2005). The Actor-Partner interdependence model: A model of bidirectional effects in developmental studies. *International Journal of Behavioral Development*, *29*(2), 101–109.

Crainiceanu, C. & Ruppert, D. (2004). Likelihood ratio tests in linear mixed models with one variance component. *Journal of the Royal Statistical Society B*, *66*, 165–185.

Curran, P. J. & Bauer, D. J. (2007). Building path diagrams for multilevel models. *Psychological Methods*, *12*(3), 283–297.

Efron, B. (1998). R. A. Fisher in the 21st Century. *Statistical Science*, 95–114.

Enders, C. K. & Bandalos, D. L. (2001). The relative performance of full information maximum likelihood estimation for missing data in structural equation models. *Structural Equation Modeling*, *8*(3), 430–457.

Estabrook, R. & Neale, M. C. (2013). A comparison of factor score estimation methods in the presence of missing data: Reliability and an application to nicotine dependence. *Multivariate Behavioral Research*, *48*(1), 1–27.

Fan, X. (1997). Canonical correlation analysis and structural equation modeling: what do they have in common? *Structural Equation Modeling: A Multidisciplinary Journal*, *4*(1), 65–79. doi:10.1080/10705519709540060

Fellner, W. H. (1987). Sparse matrices, and the estimation of variance components by likelihood methods. *Communications in Statistics-Simulation and Computation*, *16*(2), 439–463.

Gelman, A. (2005). Analysis of variance–why it is more important than ever. *The Annals of Statistics*, *33*(1), 1–53.

Gelman, A. (2008). Objections to Bayesian statistics. *Bayesian Analysis*, *3*(3), 445–449.

Gelman, A. & Shirley, K. (2011). Inference from simulations and monitoring convergence. In S. Brooks, A. Gelman, G. Jones, & X.-L. Meng (Eds.), *Handbook of Markov Chain Monte Carlo* (pp. 163–174). CRC press.

Goldstein, H. & McDonald, R. P. (1988). A general model for the analysis of multilevel data. *Psychometrika*, *53*(4), 455–467.

Hartley, H. O. & Rao, J. N. (1967). Maximum-likelihood estimation for the mixed analysis of variance model. *Biometrika*, *54*(1-2), 93–108.

Harville, D. A. (1977). Maximum likelihood approaches to variance component estimation and to related problems. *Journal of the American Statistical Association*, *72*(358), 320–338.

Henderson Jr, C. R. (1982). Analysis of covariance in the mixed model: Higher-level, nonhomogeneous, and random regressions. *Biometrics*, 623–640.

Henderson, C. R. (1953). Estimation of variance and covariance components. *Biometrics*, *9*(2), 226–252.

Jöreskog, K. G. & Van Thillo, M. (1972). LISREL: A general computer program for estimating a linear structural equation system involving multiple indicators of unmeasured variables. [Computer software].

Kenny, D. A. & Judd, C. M. (1986). Consequences of violating the independence assumption in analysis of variance. *Psychological Bulletin*, *99*(3), 422.

Lindstrom, M. J. & Bates, D. (1990). Nonlinear mixed effects models for repeated measures data. *Biometrics*, 673–687.

Lovie, P. & Lovie, A. D. (1996). Charles Edward Spearman, F.R.S. (1863–1945). *Notes and Records of the Royal Society*, *50*(1), 75–88. doi:10.1098/rsnr.1996.0007

Luenberger, D. G. & Ye, Y. (2008). *Linear and nonlinear programming*. Springer-Verlag.

MacKay, D. J. (2003). *Information theory, inference and learning algorithms*. Cambridge university press.

Maier, D. (1983). *The theory of relational databases*. Computer Science Press.

Manor, O. & Zucker, D. M. (2004). Small sample inference for the fixed effects in the mixed linear model. *Computational Statistics & Data Analysis*, *46*, 801–817. doi:10.1016/j.csda.2003.10.005

McArdle, J. J. (2005). The development of the RAM rules for latent variable structural equation modeling. In A. Maydeu-Olivares & J. J. McArdle (Eds.), *Contemporary advances in psychometrics* (pp. 225–273). Mahwah, NJ: Lawrence Erlbaum Associates, Inc.

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*, *37*(2), 234–251.

McDonald, R. P. (1978). A simple comprehensive model for the analysis of covariance structures. *British Journal of Mathematical and Statistical Psychology*, *31*(1), 59–72.

Patterson, H. D. & Thompson, R. (1971). Recovery of inter-block information when block sizes are unequal. *Biometrika*, *58*(3), 545–554.

Pawitan, Y. (2001). *In all likelihood: Statistical modelling and inference using likelihood*. Oxford University Press.

Pearl, J. (2000). *Causality: Models, reasoning and inference*. Cambridge University Press.

Pek, J. & Wu, H. (in press). Profile likelihood-based confidence intervals and regions for structural equation models. *Psychometrika*.

Pinheiro, J., Bates, D., DebRoy, S., Sarkar, D., & R Core Team. (2016). *nlme: linear and nonlinear mixed effects models*. R package version 3.1-124. Retrieved from http://CRAN.R-project.org/package=nlme

Plummer, M. (2013). *JAGS version 3.4.0 user manual*. Retrieved from http://mcmc-jags.sourceforge.net/

Pritikin, J. N. (2016, March). Data for Boker et al (2009). Open Science Framework. doi:10.17605/OSF.IO/TJQ24

Rabe-Hesketh, S., Skrondal, A., & Pickles, A. (2004). Generalized multilevel structural equation modeling. *Psychometrika*, *69*(2), 167–190.

Searle, S. R., Casella, G., & McCulloch, C. E. (1992). *Variance components*. John Wiley & Sons, Inc.

Skrondal, A. & Rabe-Hesketh, S. (2004). *Generalized latent variable modeling: Multilevel, longitudinal, and structural equation models*. CRC Press.

Stan Development Team. (2014). *Stan modeling language users guide and reference manual, version 2.5.0*. Retrieved from http://mc-stan.org/

Stanton, J. M. (2001). Galton, Pearson, and the peas: A brief history of linear regression for statistics instructors. *Journal of Statistics Education*, *9*(3).

Tarjan, R. E. (1976). Edge-disjoint spanning trees and depth-first search. *Acta Informatica*, *6*(2), 171–185.

Voelkle, M. C. & Oud, J. H. (2013). Continuous time modelling with individually varying time intervals for oscillating and non-oscillating processes. *British Journal of Mathematical and Statistical Psychology*, *66*(1), 103–126.

von Oertzen, T. & Hackett, D. C. (submitted). *Pre-processing for efficient maximum likelihood estimation in structural equation models with fixed loadings*. submitted.

West, B. T., Welch, K. B., & Galecki, A. T. (2014). *Linear mixed models: A practical guide using statistical software*. CRC Press.

White, H. (1982). Maximum likelihood estimation of misspecified models. *Econometrica: Journal of the Econometric Society*, 1–25.

Wilkinson, G. N. & Rogers, C. E. (1973). Symbolic description of factorial models for analysis of variance. *Applied Statistics*, 392–399.

Wolfinger, R., Tobias, R., & Sall, J. (1994). Computing Gaussian likelihoods and their derivatives for general linear mixed models. *SIAM Journal on Scientific Computing*, *15*(6), 1294–1310.

Appendix A

UnivariateRandomInterceptWide.R

```
#
#   Copyright 2007-2016 The OpenMx Project
#
#   Licensed under the Apache License, Version 2.0 (the "License");
#   you may not use this file except in compliance with the License.
```

```
#   You may obtain a copy of the License at
#
#        http://www.apache.org/licenses/LICENSE-2.0
#
#   Unless required by applicable law or agreed to in writing, software
#   distributed under the License is distributed on an "AS IS" BASIS,
#   WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
#   See the License for the specific language governing permissions and
#   limitations under the License.

# -----------------------------------------------------------------------------
# Program: UniRandomIntTest-120815.R
#  Author: Steve Boker
#    Date: Wed Aug 15 10:50:12 CEST 2012
#
# This program simulates some univariate multilevel data with random
# intercepts only, fits it with lme(), fits a naive wide format
# multilevel OpenMx model and checks the results
#
# -----------------------------------------------------------------------------
# Revision History
#   Steve Boker -- Wed Aug 15 10:50:14 CEST 2012
#      Created UniRandomIntTest-120815.R
#
# -----------------------------------------------------------------------------

# ----------------------------------
# Read libraries and set options.

options(width=110)
library(nlme)
library(OpenMx)

# ----------------------------------
# Set constants.

sdLevelOneE <- sqrt(.2)
sdIntercepts <- sqrt(.5)
sdX <- sqrt(1)

N <- 400     # number of participants
P <- 100     # number of observations per participant
b0 <- .5     # Fixed effect intercept
```

```
b1 <- .8     # Fixed effect slope

set.seed(1)

# ---------------------------------------------------------
# Simulate the data.

X <- rnorm(N*P, 0, sd=sdX)
ID <- rep(1:N, each=P)
b0i <- b0 + rnorm(N, 0, sd=sdIntercepts)
Y <- rep(b0i, each=P) + b1*X + rnorm(N*P, 0, sd=sdLevelOneE)

SimUniRandomIntFrame <- data.frame(ID, X, Y)

# ---------------------------------------------------------
# Test with lme().

lmeOut <- summary(lme(Y ~ X, random= list(~ 1 | ID),
                            data=SimUniRandomIntFrame))

# For lme4, use:
# lmerOut <- lmer(Y ~ X + (1 | ID), data=SimUniRandomIntFrame)

# ---------------------------------------------------------
# Set constants.

theIDs <- unique(SimUniRandomIntFrame$ID)
totalN <- length(theIDs)
totalVars <- 2

maxP <- 0
for (tID in theIDs) {
    tmask <- SimUniRandomIntFrame$ID==tID
    tLen <- length(SimUniRandomIntFrame$ID[tmask])
    if (tLen > maxP)
        maxP <- tLen
}

# ---------------------------------------------------------
# Wide-format the data frame from tall format.

wideMatrix <- matrix(NA, nrow=totalN, ncol=1 + (maxP*totalVars))
colnames(wideMatrix) <- c("ID", paste("Y",1:maxP, sep=""),
```

```
                          paste("X",1:maxP, sep=""))
i <- 1
for (tID in theIDs) {
    wideMatrix[i, 1] <- tID
    tY <- SimUniRandomIntFrame$Y[SimUniRandomIntFrame$ID==tID]
    wideMatrix[i, 2:(length(tY)+1)] <- tY
    tX <- SimUniRandomIntFrame$X[SimUniRandomIntFrame$ID==tID]
    wideMatrix[i, (2+maxP):(length(tY)+1+maxP)] <- tX
    i <- i + 1
}
wideFrame <- data.frame(wideMatrix)

manifestNames <- colnames(wideFrame)[2:dim(wideFrame)[2]]
xNames <- paste("X",1:maxP, sep="")
yNames <- paste("Y",1:maxP, sep="")
latentNames <- c("b0i")

# ----------------------------------------
# Build the OpenMx wide model.

OpenMxModelUniRandomIntModel1 <-
  mxModel("OpenMxModelUniRandomIntModel1",
        type="RAM",
        manifestVars=manifestNames,
    latentVars=latentNames,
    mxPath(from=xNames, to=yNames, connect="single", arrows=1,
           free=TRUE, values=.2, labels="b1"),
    mxPath(from=xNames, to=xNames, connect="single", arrows=2,
           free=TRUE, values=.8, labels="vX"),
    mxPath(from=yNames, to=yNames, connect="single", arrows=2,
           free=TRUE, values=.8, labels="eY"),
    mxPath(from=latentNames, to=yNames, arrows=1, free=FALSE, values=1),
    mxPath(from=latentNames, to=latentNames, connect="single", arrows=2,
           free=TRUE, values=.8, labels="vb0i"),
    mxPath(from="one", to=c(xNames), arrows=1,
           free=TRUE, values=1, labels="mX"),
    mxPath(from="one", to=c(latentNames), arrows=1,
           free=TRUE, values=1, labels="mb0i"),
    mxData(observed=wideFrame, type="raw")
)

# ----------------------------------------
# Fit the model and examine the summary results.
```

```

omxFit <- mxRun(OpenMxModelUniRandomIntModel1)

summary(omxFit)

omxCheckCloseEnough(lmeOut$coefficients$fixed[1],
                    mxEval(mb0i, model=omxFit), 0.001)

omxCheckCloseEnough(lmeOut$coefficients$fixed[2],
                    mxEval(b1, model=omxFit), 0.001)

omxCheckCloseEnough(lmeOut$sigma,
                    mxEval(sqrt(eY), model=omxFit), 0.001)

omxCheckCloseEnough(sd(c(lmeOut$coefficients$random$ID)),
                    mxEval(sqrt(vb0i), model=omxFit), 0.001)

if (0) {
        omxCheckCloseEnough(lmeOut$coefficients$fixed,
                            fixef(lmerOut), 1e-4)
  omxCheckCloseEnough(lmeOut$sigma, sigma(lmerOut), 1e-4)
  omxCheckCloseEnough(c(lmeOut$coefficients$random$ID),
                      ranef(lmerOut)$ID[[1]], 1e-4)
}
```

Appendix B

lmer sleepstudy example

```
library(lme4)
fm1 <- lmer(Reaction ~ Days + (Days | Subject), sleepstudy, REML=FALSE)

library(OpenMx)

if (is.factor(sleepstudy$Subject)) {
  subjnum <- unclass(sleepstudy$Subject)
  sleepstudy$Subject <- as.integer(levels(sleepstudy$Subject)[ subjnum ])
}

bySubj <- mxModel(
    model="bySubj", type="RAM",
    latentVars=c("slope", "intercept"),
    mxData(data.frame(Subject=unique(sleepstudy$Subject)),
           type="raw", primaryKey = "Subject"),
```

```
    mxPath(from=c("intercept", "slope"), arrows=2, values=1),
    mxPath(from="intercept", to="slope", arrows=2, values=.25, labels="cov1"))

sleepModel <- mxModel(
    model="sleep", type="RAM", bySubj,
    manifestVars="Reaction", latentVars = "Days",
    mxData(sleepstudy, type="raw", sort=FALSE),
    mxPath(from="one", to="Reaction", arrows=1, free=TRUE),
    mxPath(from="one", to="Days", arrows=1, free=FALSE, labels="data.Days"),
    mxPath(from="Days", to="Reaction", arrows=1, free=TRUE),
    mxPath(from="Reaction", arrows=2, values=1),
    mxPath(paste0('bySubj.', c('intercept','slope')),
           'Reaction', arrows=1, free=FALSE, values=c(1,NA),
           labels=c(NA, "data.Days"), joinKey="Subject"))

m1 <- mxRun(sleepModel)

omxCheckCloseEnough(logLik(m1), logLik(fm1), 1e-6)
```

Appendix C

lmer Orthodont example

```
libraries <- rownames(installed.packages())
if (!all(c("lme4","nlme") %in% libraries)) stop("SKIP")

library(lme4)
data(Orthodont, package="nlme")
Orthodont$nsex <- as.numeric(Orthodont$Sex=="Male")
Orthodont$nsexage <- with(Orthodont, nsex*age)
fm1 <- lmer(distance ~ age + (age|Subject) + (0+nsex|Subject) +
                (0 + nsexage|Subject), data=Orthodont, REML=FALSE)

library(OpenMx)

if (is.factor(Orthodont$Subject)) {
    Orthodont$Subject <- as.integer(unclass(Orthodont$Subject))
}

bySubj <- mxModel(
    model="subj", type="RAM",
    latentVars = c('intercept', paste0(c("age", 'nsex', "nsexage"), "L")),
    mxData(data.frame(Subject=unique(Orthodont$Subject)),
           type="raw", primaryKey="Subject"),
```

```
    mxPath(from=c('intercept', 'ageL'), to=c('intercept', 'ageL'),
           arrows=2, "unique.pairs", values=c(1,.1,1),
           labels=c('subjInt', 'subjIntAge', 'subjAge')),
    mxPath(from=c('nsexL', 'nsexageL'), arrows=2, values=1))

ortho <- mxModel(
    model="ortho", bySubj, type="RAM", manifestVars=c("distance"),
    latentVars = c("ageL"),
    mxData(type="raw", observed=Orthodont[,c('distance', 'age',
                          'Subject', 'nsex', "nsexage")], sort = FALSE),
    mxPath(from=c("one"), to="distance"),
    mxPath(from=c("one"), to="ageL", free=FALSE, labels="data.age"),
    mxPath(from="ageL", to="distance"),
    mxPath(from="distance", arrows=2, values=1),
    mxPath(from="subj.intercept", to="distance", values=1, free=FALSE,
           joinKey="Subject"),
    mxPath(from=paste0("subj.", c("ageL", "nsexL", "nsexageL")),
           to="distance",
           labels=paste0("data.", c("age", "nsex", "nsexage")),
           free=FALSE, joinKey="Subject"))

if (1) {
  # load lme4's parameters
    p1 <- ortho
    p1$subj$S$values[c('intercept', 'ageL'),c('intercept', 'ageL')] <-
        VarCorr(fm1)$Subject
    p1$subj$S$values[c('nsexL'),c('nsexL')] <-
        VarCorr(fm1)$Subject.1
    p1$subj$S$values[c('nsexageL'),c('nsexageL')] <-
        VarCorr(fm1)$Subject.2

    p1$A$values['distance','ageL'] <- fixef(fm1)['age']
    p1$M$values[,'distance'] <- fixef(fm1)['(Intercept)']
    p1$S$values['distance','distance'] <- getME(fm1, "sigma")^2

    pt1 <- mxRun(mxModel(p1, mxComputeSequence(list(
        mxComputeOnce('fitfunction', 'fit'),
        mxComputeReportExpectation()))))

    omxCheckCloseEnough(logLik(pt1), logLik(fm1), 1e-6)
}

orthoFit <- mxRun(ortho)
```

```

# OpenMx finds a better solution
omxCheckCloseEnough(orthoFit$output$fit, 436.73, 1e-2)

# ------------------------------------------------------------

fm2 <- lmer(distance ~ age + (age|Subject) + (0+nsex|Subject) +
                (0 + nsexage|Subject), data=Orthodont, REML=TRUE)

ortho$fitfunction$profileOut <- c("ortho.A[1,2]", "ortho.M[1,1]")

if (1) {
  # load lme4's parameters
    p1 <- ortho
    p1$subj$S$values[c('intercept', 'ageL'),c('intercept', 'ageL')] <-
        VarCorr(fm2)$Subject
    p1$subj$S$values[c('nsexL'),c('nsexL')] <-
        VarCorr(fm2)$Subject.1
    p1$subj$S$values[c('nsexageL'),c('nsexageL')] <-
        VarCorr(fm2)$Subject.2

    p1$A$values['distance','ageL'] <- fixef(fm2)['age']
    p1$M$values[,'distance'] <- fixef(fm2)['(Intercept)']
    p1$S$values['distance','distance'] <- getME(fm2, "sigma")^2

    pt1 <- mxRun(mxModel(p1, mxComputeSequence(list(
        mxComputeOnce('fitfunction', 'fit'),
        mxComputeReportExpectation()))))

    omxCheckCloseEnough(logLik(pt1), logLik(fm2), 1e-6)
}

orthoFit <- mxRun(ortho)

omxCheckCloseEnough(orthoFit$output$fit, 440.43, .01)
```

Appendix D

MultilevelUniRandomSlopeInt.R

```
#
#   Copyright 2007-2016 The OpenMx Project
#
#   Licensed under the Apache License, Version 2.0 (the "License");
```

```
#   you may not use this file except in compliance with the License.
#   You may obtain a copy of the License at
#
#        http://www.apache.org/licenses/LICENSE-2.0
#
#   Unless required by applicable law or agreed to in writing, software
#   distributed under the License is distributed on an "AS IS" BASIS,
#   WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
#   See the License for the specific language governing permissions and
#   limitations under the License.

require(OpenMx)
require(nlme)

# Multilevel Long Format Test
# Author: Steve Boker
# Date: Sun Nov 29 14:06:07 EST 2009


# This script is used to test the multilevel long format
# functionality using definition variables as indices.
totalOccasions <- 100
totalSubjects <- 10L
set.seed(42) # repeatibility
tID <- rep(1:totalSubjects, each=totalOccasions)
trueX <- rep(rnorm(totalOccasions, mean=0, sd=2), each=totalSubjects) +
    rnorm(totalOccasions*totalSubjects, mean=0, sd=.2)
trueB <- rep(rnorm(totalSubjects, mean=.8, sd=.3), each=totalOccasions)
tDataFrame <- data.frame(
    ID=tID, X=trueX, Y=trueB*trueX +
        rnorm(totalOccasions*totalSubjects,mean=0, sd=.1),trueB=trueB)
summary(tDataFrame)

manifestVars <- c("X", "Y")
numSubjects <- length(unique(tDataFrame$ID))

# Estimates the sum of the random and fixed effects
multilevelModel2 <- mxModel("Multilevel_2",
    mxMatrix("Full", nrow=numSubjects, ncol=2,
        values=c(.2,0),
        free=c(TRUE, TRUE),
        name="Rand",
        byrow=TRUE
```

```
    ),
    mxMatrix("Full", 2, 2,
        labels=c(NA,   NA,
                "randrow[1,1]", NA),
        free=FALSE,
        name="A",
        byrow=TRUE
    ),
    mxMatrix("Symm", 2, 2,
        values=c(.9,0,.9),
        free=c(T,
               F, T),
        labels=c("varX",
                 NA, "varY"),
        name="S",
        byrow=TRUE
    ),
        mxMatrix("Full", 2, 2,
                values=c(1,0,
                              0,1),
                    free=FALSE,
                    byrow=TRUE, name="F"),
    mxMatrix("Iden", 2, name="I"),
    mxAlgebra(F %*% solve(I-A) %*% S %*% t(solve(I-A)) %*% t(F),
        name="R",
        dimnames = list(manifestVars, manifestVars)
    ),
    mxMatrix("Full", nrow=1, ncol=length(manifestVars),
        values=0,
        free=FALSE,
        labels=c(NA,"randrow[1,2]"),
        dimnames=list(NULL, manifestVars),
        name="M"
    ),
        mxAlgebra(Rand[data.ID,], name="randrow"),
    mxFitFunctionML(),mxExpectationNormal(covariance="R", means="M"),
    mxData(tDataFrame, type="raw")
)

# -----------------------------------------------------------------
# Fit the model and examine the summary results.

multilevelModel2Fit <- mxRun(multilevelModel2)
```

```
summary(multilevelModel2Fit)

lmeOut <- lme(Y~X, random= ~ X | ID, data=tDataFrame)

cbind(multilevelModel2Fit$output$estimate[1:numSubjects],
      lmeOut$coef$random$ID[,2] + lmeOut$coef$fixed[2],
      trueB[seq(1,totalOccasions*(totalSubjects), by=totalOccasions)])

mean(multilevelModel2Fit$output$estimate[1:numSubjects])

est <- multilevelModel2Fit$output$estimate

omxCheckCloseEnough(mean(est[1:numSubjects]),
    lmeOut$coef$fixed[2], 0.001)

omxCheckCloseEnough(mean(est[(1:numSubjects) + (1*numSubjects)]),
    lmeOut$coef$fixed[1], 0.001)

# ----------------------------------------------------------------
# An OpenMx equivalent to the mixed model

perID <- mxModel(
    "perID", type="RAM", latentVars=c('int', 'slope'),
    mxData(data.frame(ID=1L:totalSubjects), "raw", primaryKey="ID"),
    mxPath(c('int', 'slope'),c('int', 'slope'),'unique.pairs',
           arrows=2,values=c(1,0,1)))

occa <- mxModel(
    "occa", type="RAM", perID, manifestVars="Y", latentVars="lX",
    mxData(tDataFrame, 'raw', sort=FALSE),
    mxPath('Y', arrows=2, values=1),
    mxPath('one', 'Y'),
    mxPath('one', 'lX', labels='data.X', free=FALSE),
    mxPath('lX', 'Y'),
    mxPath('perID.int', 'Y', values=1, free=FALSE, joinKey='ID'),
    mxPath('perID.slope', 'Y', labels='data.X', free=FALSE, joinKey='ID'))

if (0) {
        require(lme4)
        lmer1 <- lmer(Y~X + (X | ID), data=tDataFrame, REML=FALSE)
        pt1 <- occa
        #pt1$perID$cholS$values[,] <- chol(VarCorr(lmer1)$ID)
```

```
        pt1$perID$S$values[,] <- VarCorr(lmer1)$ID
        pt1$A$values['Y', 'lX'] <- fixef(lmer1)['X']
        pt1$M$values[,'Y'] <- fixef(lmer1)['(Intercept)']
        pt1$S$values['Y', 'Y'] <- getME(lmer1, "sigma")^2

        pt1 <- mxRun(mxModel(pt1, mxComputeSequence(list(
            mxComputeOnce('fitfunction', 'fit'),
            mxComputeReportExpectation()))))

        omxCheckCloseEnough(logLik(pt1), logLik(lmer1), 1e-6)
}

occa <- mxRun(occa)
# a tad better than lme, same as lmer
omxCheckCloseEnough(occa$output$fit, -1725.954, 1e-2)
```

Appendix E

Rampart proof-of-concept test script ported from June 2013 prototype

```
# This is the original test case that Timo & I wrote back in Spring 2013.

#options(error = utils::recover)   # uncomment for more help with debugging
library(OpenMx)
library(mvtnorm)

set.seed(1)

more.noise <- 0
#more.noise <- 1

gen.data <- function(n) {
  data.cov <- matrix(c(1, .2, .2, 1), byrow=TRUE, nrow=2)
  latent <- rmvnorm(n, mean=c(0,0), sigma=data.cov)
  colnames(latent) <- c("A","B")
  latent <- as.data.frame(latent)
  df <- data.frame(C=latent$A + latent$B,
                   D=latent$A - latent$B)
  if (more.noise) {
    df$C <- df$C + rnorm(1, sd=more.noise)
    df$D <- df$D + rnorm(1, sd=more.noise)
  }
  df
}
```

```

fanout <- 5

school.data <- cbind(id=1:fanout, gen.data(fanout))
#school.data$C <- school.data$id * 1000
teacher.data <- cbind(schoolId=1:fanout, id=seq(1,fanout^2),
                      gen.data(fanout^2))
#teacher.data$C <- teacher.data$id * 100
student.data <- cbind(teacherId=seq(1,fanout^2),
                      id=seq(1,fanout^3), gen.data(fanout^3))

stack.data <- function(key, upper, lower) {
        for (pk in upper$id) {
                mask <- lower[[key]] == pk
                for (col in c('C','D')) {
                        lower[mask, col] <-
                                lower[mask, col] + upper[upper$id == pk, 'C']
                }
        }
        lower
}
teacher.data <- stack.data("schoolId", school.data, teacher.data)
student.data <- stack.data("teacherId", teacher.data, student.data)

manifests<-c("C","D")
latents<-c("A","B")
student <- mxModel(
    "student", type="RAM",
    manifestVars = manifests,
    latentVars = latents,
    mxPath(from="A",to=c("C","D"), free=c(FALSE,FALSE),
           value=c(1,1), arrows=1,
           label=c("A_TO_C","A_TO_D") ),
    mxPath(from="B",to=c("C","D"), free=c(FALSE,FALSE), value=c(1,-1) ,
           arrows=1, label=c("B_TO_C","B_TO_D") ),
    mxPath(from="A",to=c("A","B"), free=c(TRUE,TRUE),
           value=c(1,0), arrows=2,
           label=c("VAR_A","COV_A_B") ),
    mxPath(from="B",to=c("B"), free=c(TRUE), value=c(1) , arrows=2,
           label=c("VAR_B") ),
    mxPath(from="C",to=c("C"), free=as.logical(more.noise),
           value=more.noise, arrows=2, label=c("VAR_C") ),
    mxPath(from="D",to=c("D"), free=as.logical(more.noise),
```

```
            value=more.noise, arrows=2, label=c("VAR_D") ),
    mxPath(from="one", to=c(manifests, latents), value=0, free=FALSE)
);

relabel <- function(m, prefix) {
  for (mat in c("A","S")) {
    lab <- m[[mat]]$labels
    lab[!is.na(lab)] <- paste0(prefix, lab[!is.na(lab)])
    m[[mat]]$labels <- lab
  }
  m
}

teacher <- relabel(mxModel(student, name="teacher"), "tea_")
school <- relabel(mxModel(student, name="school"), "sch_")
student <- relabel(student, "st_")

school <- mxModel(
    school,
    mxData(school.data, type="raw", primaryKey="id", sort=FALSE))

teacher <- mxModel(
    teacher, school,
    mxData(teacher.data, type="raw", primaryKey="id", sort=FALSE),
    mxPath('school.C', 'A', free=FALSE, value=1, joinKey="schoolId"))

student <- mxModel(
    student, teacher,
    mxData(student.data, type="raw", primaryKey="id", sort=FALSE),
    mxPath('teacher.C', 'A', free=FALSE, value=1, joinKey="teacherId"))

#student$expectation$verbose <- 1L

student$expectation$.rampart <- 0L
pt1 <- mxRun(mxModel(
    student,
    mxComputeSequence(list(
        mxComputeOnce('fitfunction', 'fit'),
        mxComputeNumericDeriv(checkGradient=FALSE,
                              iterations=2, hessian=FALSE),
        mxComputeReportDeriv(),
        mxComputeReportExpectation()))))

```

```
student$expectation$.rampart <- as.integer(NA)
pt2 <- mxRun(mxModel(
    student,
    mxComputeSequence(list(
        mxComputeOnce('fitfunction', 'fit'),
        mxComputeNumericDeriv(checkGradient=FALSE,
                              iterations=2, hessian=FALSE),
        mxComputeReportDeriv(),
        mxComputeReportExpectation()))))

omxCheckCloseEnough(pt2$expectation$debug$rampartUsage,
                    c((fanout-1)*fanout^2, (fanout-1)*fanout), 1)
omxCheckCloseEnough(pt2$expectation$debug$numGroups, 3)

if (0) {
        layout <- pt2$expectation$debug$layout
        head(layout[layout$group==3, ],n=20)
}

omxCheckCloseEnough(pt1$output$fit, pt2$output$fit, 1e-7)
omxCheckCloseEnough(pt1$output$gradient, pt2$output$gradient, 1e-6)

student <- mxRun(student)
if (!more.noise) {
        omxCheckCloseEnough(student$output$fit, 1055.161, 1e-2)
} else {
        omxCheckCloseEnough(student$output$fit, 1132.713, 1e-2)  # but code RED
}
#print(student$expectation$debug$rampartUsage)

if (0) {
        ex <- student$expectation
        eo = ex$output
        ed = ex$debug
        ed$layout
}

got <- mxGenerateData(student)
omxCheckEquals(names(got), c("school", "teacher", "student"))
omxCheckEquals(colnames(got[['school']]),
               colnames(student$school$data$observed))
omxCheckTrue(all(got[['school']]$C != student$school$data$observed$C))
```

```
omxCheckError(mxGenerateData(student, 10, returnModel=TRUE),
              paste("Specification of the number of rows",
                    "is not supported for relational models"))

got <- mxGenerateData(student, returnModel=TRUE)
omxCheckTrue(is(got, "MxModel"))
omxCheckTrue(all(got$school$data$observed$C != student$school$data$observed$C))
```

Appendix F

univACErSEM.R

```
#
#   Copyright 2007-2016 The OpenMx Project
#
#   Licensed under the Apache License, Version 2.0 (the "License");
#   you may not use this file except in compliance with the License.
#   You may obtain a copy of the License at
#
#        http://www.apache.org/licenses/LICENSE-2.0
#
#   Unless required by applicable law or agreed to in writing, software
#   distributed under the License is distributed on an "AS IS" BASIS,
#   WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
#   See the License for the specific language governing permissions and
#   limitations under the License.
#------------------------------------------------------------------------------

#------------------------------------------------------------------------------
# Author: Michael D. Hunter
# Date: 2016-02-03
# Filename: univACErSEM.R
# Purpose: Define a behavior genetics single-trait ACE model as a
#  Relational SEM (rSEM)
#------------------------------------------------------------------------------


#------------------------------------------------------------------------------
require(OpenMx)


#------------------------------------------------------------------------------
# Prepare Data

```

```r
data("twinData", package="OpenMx")
selVars <- c('bmi1','bmi2','zyg')
wideData <- subset(twinData, zyg %in% c(1, 3), selVars)
wideData$rel <- c(1, NA, .5)[wideData$zyg]
wideData$famID <- 1:nrow(wideData)
tallData <- reshape(wideData, varying=c('bmi1', 'bmi2'), v.names='bmi',
                    timevar='twin', times=1:2, idvar='famID', direction='long')
tallData$personID <- 1:nrow(tallData)
tallData$relsqrt <- sqrt(tallData$rel)
tallData$relu <- sqrt(1-tallData$rel)
tallData <- tallData[order(tallData$famID, tallData$twin),
                     c('famID', 'personID', 'twin', 'rel',
                       'relsqrt', 'relu', 'bmi')]
wData <- tallData
bData <- tallData[!duplicated(tallData$famID),
                  c('famID', 'rel', 'relsqrt')]


#----------------------------------------------------------------------------------
# Between Model

bModel <- mxModel(
    'between', type="RAM",
    mxData(type="raw", observed=bData, primaryKey="famID"),
    latentVars = c("C", "AC"),
    mxPath("C", arrows=2, values=1, labels="v_C", lbound=1e-6),
    mxPath("AC", arrows=2, values=1, labels="v_A", lbound=1e-6))


#----------------------------------------------------------------------------------
# Within Model

wModel <- mxModel(
    'within', type="RAM", bModel,
    mxData(type="raw", observed=wData, sort=FALSE),
    manifestVars = 'bmi',
    latentVars = c("E", "AU"),
    mxPath(from="one", to="bmi", arrows=1, free=TRUE, values=20, labels="mean"),
    mxPath('E', arrows=2, values=1, labels="v_E", lbound=1e-6),
    mxPath('AU', arrows=2, values=1, labels="v_A", lbound=1e-6),
    mxPath('AU', 'bmi', values=1, labels='data.relu', free=FALSE),
    mxPath('E', 'bmi', free=FALSE, values=1),
    mxPath('between.C', 'bmi', values=1,
```

```
             free=FALSE, joinKey="famID"),
    mxPath('between.AC', 'bmi', values=1, arrows=1, free=FALSE,
             labels='data.relsqrt', joinKey="famID"))


#------------------------------------------------------------------------------
# Run 'em
wRun <- mxRun(wModel)


#------------------------------------------------------------------------------
# Take a look

summary(wRun)

# Cf. inst/models/passing/univACEP.R

#Mx answers hard-coded
#1: Heterogeneity Model
Mx.A <- 0.6173023
Mx.C <- 5.595822e-14
Mx.E <- 0.1730462
Mx.M <- 21.39293
Mx.LL_ACE <- 4067.663

wparam <- mxEval(rbind(v_A, v_C, v_E, mean), wRun)
mparam <- rbind(Mx.A, Mx.C, Mx.E, Mx.M)
omxCheckCloseEnough(wparam, mparam, .001)

omxCheckCloseEnough(-2*logLik(wRun), Mx.LL_ACE, .001)


#------------------------------------------------------------------------------
# Same model, but with constant between-level transition matrix

bLatent <- c('C', 'AC')
bModel2 <- mxModel(
    'between',
    mxData(type="raw", observed=bData, primaryKey="famID"),
    latentVars = bLatent,
    mxMatrix(name="F", nrow=0, ncol=2, dimnames=list(NULL, bLatent)),
    mxAlgebra(data.rel * v_A, name="rel_v_A"),
    mxMatrix("Symm", name="S", nrow=2, ncol=2, dimnames=list(bLatent,bLatent),
```

```
                  free=c(TRUE,FALSE,FALSE), labels=c("v_C", NA, "rel_v_A[1,1]"),
                  values=c(1,0,1), lbound=c(1e-6,NA,1e-6)),
        mxMatrix(name="A", nrow=2, ncol=2, values=0,
                  dimnames=list(bLatent,bLatent)),
        mxFitFunctionML(),
        mxExpectationRAM())

#--------------------------------------------------------------------------------
# Within Model

wModel2 <- mxModel(
        'within', type="RAM", bModel2,
        mxData(type="raw", observed=wData, sort=FALSE),
        manifestVars = 'bmi',
        latentVars = c("E", "AU"),
        mxPath(from="one", to="bmi", arrows=1, free=TRUE,
               values=20, labels="mean"),
        mxPath('E', arrows=2, values=1, labels="v_E", lbound=1e-6),
        mxPath('AU', arrows=2, values=1, labels="v_A", lbound=1e-6),
        mxPath('AU', 'bmi', values=1, labels='data.relu', free=FALSE),
        mxPath('E', 'bmi', free=FALSE, values=1),
        mxPath('between.C', 'bmi', values=1,
               free=FALSE, joinKey="famID"),
        mxPath('between.AC', 'bmi', values=1,
               free=FALSE, joinKey="famID"))

# This isn't a huge speed-up because the per-cluster covariance matrix
# is already small in the version above.
wRun2 <- mxRun(wModel2)

wparam <- mxEval(rbind(v_A, v_C, v_E, mean), wRun2)
mparam <- rbind(Mx.A, Mx.C, Mx.E, Mx.M)
omxCheckCloseEnough(wparam, mparam, .001)

omxCheckCloseEnough(-2*logLik(wRun2), Mx.LL_ACE, .001)

omxCheckCloseEnough(wRun2$expectation$debug$rampartUsage, 867, 1)
```

Appendix G

mplus-ex9.6.R

```
# MPLUS: TWO-LEVEL CFA WITH CONTINUOUS FACTOR INDICATORS AND COVARIATES
# See https://www.statmodel.com/usersguide/chapter9.shtml
```

```
library(OpenMx)

set.seed(1)
ex96 <- suppressWarnings(try(read.table("models/nightly/data/ex9.6.dat")))
if (is(ex96, "try-error")) ex96 <- read.table("data/ex9.6.dat")

ex96$V8 <- as.integer(ex96$V8)
bData <- ex96[!duplicated(ex96$V8), c('V7', 'V8')]
colnames(bData) <- c('w', 'clusterID')
wData <- ex96[,-match(c('V7'), colnames(ex96))]
colnames(wData) <- c(paste0('y', 1:4), paste0('x', 1:2), 'clusterID')

bModel <- mxModel(
    'between', type="RAM",
    mxData(type="raw", observed=bData, primaryKey="clusterID"),
    latentVars = c("lw", "fb"),
    mxPath("one", "lw", labels="data.w", free=FALSE),
    mxPath("fb", arrows=2, labels="psiB"),
    mxPath("lw", 'fb', labels="phi1"))

wModel <- mxModel(
    'within', type="RAM", bModel,
    mxData(type="raw", observed=wData, sort=FALSE),
    manifestVars = paste0('y', 1:4),
    latentVars = c('fw', paste0("xe", 1:2)),
    mxPath("one", paste0('y', 1:4), values=runif(4),
           labels=paste0("gam0", 1:4)),
    mxPath("one", paste0('xe', 1:2),
           labels=paste0('data.x',1:2), free=FALSE),
    mxPath(paste0('xe', 1:2), "fw",
           labels=paste0('gam', 1:2, '1')),
    mxPath('fw', arrows=2, values=1.1, labels="varFW"),
    mxPath('fw', paste0('y', 1:4), free=c(FALSE, rep(TRUE, 3)),
           values=c(1,runif(3)), labels=paste0("loadW", 1:4)),
    mxPath('between.fb', paste0('y', 1:4), values=c(1,runif(3)),
           free=c(FALSE, rep(TRUE, 3)), labels=paste0("loadB", 1:4),
           joinKey="clusterID"),
    mxPath(paste0('y', 1:4), arrows=2, values=rlnorm(4),
           labels=paste0("thetaW", 1:4)))

mle <- structure(c(
    0.9989, 0.9948, 1.0171, 0.9809, 0.9475, 1.0699,
```

```r
    1.0139, 0.9799, -0.0829, -0.0771, -0.0449, -0.0299, 0.9728, 0.5105,
    0.9595, 0.9238, 0.9489, 0.361, 0.3445),
                .Names = c("loadW2", "loadW3", "loadW4", "thetaW1",
                    "thetaW2", "thetaW3", "thetaW4", "varFW",
                    "gam01", "gam02", "gam03", "gam04", "gam11", "gam21",
                    "loadB2", "loadB3", "loadB4", "psiB", "phi1"))

if (1) {
        pt1 <- omxSetParameters(wModel, labels=names(mle), values=mle)
#       pt1$expectation$.forceSingleGroup <- TRUE
#       pt1$expectation$.rampart <- 0L
        plan <- mxComputeSequence(list(
            mxComputeOnce('fitfunction', 'fit'),
#           mxComputeNumericDeriv(checkGradient=FALSE,
#                                 hessian=FALSE, iterations=2),
            mxComputeReportDeriv(),
            mxComputeReportExpectation()
        ))
        pt1 <- mxRun(mxModel(pt1, plan))
        omxCheckCloseEnough(pt1$output$fit, 13088.373, 1e-2)
}

if (1) {
#  wModel <- mxRun(mxModel(wModel, mxComputeGradientDescent(verbose=2L)))
  wModel <- mxRun(wModel)
  summary(wModel)

  omxCheckCloseEnough(wModel$output$fit, 13088.373, 1e-2)
  omxCheckCloseEnough(mle[names(coef(wModel))], coef(wModel), 1e-3)
  omxCheckCloseEnough(wModel$expectation$debug$rampartUsage, 890)
} else {
        options(width=120)
        plan <- mxComputeSequence(list(
            mxComputeOnce('fitfunction', 'fit'),
            mxComputeNumericDeriv(checkGradient=FALSE,
                                  hessian=FALSE, iterations=2),
            mxComputeReportDeriv(),
            mxComputeReportExpectation()
        ))

        wModel$expectation$.rampart <- 2L
#       wModel$expectation$scaleOverride <- c(6, 1)
        rotated <- mxRun(mxModel(wModel, plan))
```

```
        wModel$expectation$.rampart <- 0L
        square <- mxRun(mxModel(wModel, plan))

        ex <- rotated$expectation
        eo <- ex$output
        ed <- ex$debug
        print(ed$rampartUsage)
        print(abs(rotated$output$fit - square$output$fit))
        print(max(abs(rotated$output$gradient - square$output$gradient)))
}
```

Appendix H

multilevelLatentRegression2.R

```
library(OpenMx)

set.seed(1)

numIndicators <- 4

numDistricts <- 5
numSchools <- 4
numTeachers <- 3
numStudents <- 5

genData <- function(upper, fanout, keyname) {
        lowerData <- NULL
        for (sx in 1:nrow(upper)) {
                extraFanout <- sample.int(fanout, 1)
#               extraFanout <- 0L
                lowerData <- rbind(lowerData, data.frame(
                    upper=upper[sx,1], skill=rnorm(fanout + extraFanout,
                                              mean=upper[sx, 'skill'])))
        }
        colnames(lowerData)[[1]] <- colnames(upper)[[1]]
        lowerData[[keyname]] <- 1:nrow(lowerData)
        lowerData <- lowerData[,c(3,1,2)]
        lowerData
}

districtData <- data.frame(districtID=1:numDistricts,
                           skill=rnorm(numDistricts))
```

```
schoolData <- genData(districtData, numSchools, 'schoolID')
teacherData <- genData(schoolData, numTeachers, 'teacherID')
studentData <- genData(teacherData, numStudents, 'studentID')

createIndicators <- function(latentSkill, indicatorVariance) {
        if (missing(indicatorVariance)) {
                indicatorVariance <- rep(1, numIndicators)
                                        #rlnorm(numIndicators) / 8
        }
        ind <- matrix(NA, length(latentSkill), length(indicatorVariance))
        for (ix in 1:length(latentSkill)) {
                ind[ix,] <-
                  sapply(indicatorVariance,
                         function(sd) rnorm(1, mean=latentSkill[ix], sd=sd))
        }
        # per indicator mean
#       ind <- t(t(ind) + runif(numIndicators,min=-1,max=1))
        colnames(ind) <- paste0('i', 1:length(indicatorVariance))
        as.data.frame(ind)
}

districtData <- cbind(districtData, createIndicators(districtData$skill))
schoolData <- cbind(schoolData, createIndicators(schoolData$skill))
teacherData <- cbind(teacherData, createIndicators(teacherData$skill))
studentData <- cbind(studentData, createIndicators(studentData$skill))

studentData$i4[runif(nrow(studentData)) > .8] <- NA
#teacherData$i4[runif(nrow(teacherData)) > .8] <- NA

mkSingleFactor <- function(latent=c()) {
        mxModel('template', type='RAM',
                manifestVars = paste0('i', 1:numIndicators),
                latentVars = c("skill",latent),
                mxPath(from='skill', arrows=2, labels="Var",
                       values=rlnorm(1), lbound=.01),
                mxPath(from=paste0('i',1:numIndicators), arrows=2,
                       values=rlnorm(1), labels="Err", lbound=.01),
                mxPath(from="one", to=paste0('i',1:numIndicators),
                       free=TRUE, values=rnorm(4)),
                mxPath(from='skill', to=paste0('i',1:numIndicators),
                       labels=paste0('L',1:numIndicators), lbound=0,
                       values=c(1, runif(numIndicators-1, .5,1.5)),
                       free=c(FALSE, rep(TRUE,numIndicators-1)))
```

```
                  )
}

singleFactor <- mkSingleFactor(NULL)

relabel <- function(m, prefix) {
  for (mat in c("A","S")) {
    lab <- m[[mat]]$labels
    lab[!is.na(lab)] <- paste0(prefix, lab[!is.na(lab)])
   m[[mat]]$labels <- lab
  }
  mxModel(m, name=prefix)
}

dMod <- mxModel(relabel(mkSingleFactor(), "district"),
                mxData(type="raw", observed=districtData,
                       primaryKey="districtID", sort=FALSE))

schMod <- mxModel(relabel(mkSingleFactor(), "school"), dMod,
                  mxData(type="raw", observed=schoolData,
                         primaryKey="schoolID", sort=FALSE),
                  mxPath(from='district.skill', to='skill',
                         joinKey="districtID", values=runif(1)))

tMod <- mxModel(relabel(singleFactor, "teacher"), schMod,
                mxData(type="raw", observed=teacherData,
                       primaryKey="teacherID", sort=FALSE),
                  mxPath(from='school.skill', to='skill',
                         joinKey="schoolID", values=runif(1)))

sMod <- mxModel(relabel(singleFactor, "student"), tMod,
                  mxData(type="raw", observed=studentData,
                         primaryKey="studentID", sort=FALSE),
                  mxPath(from='teacher.skill', to='skill',
                         joinKey="teacherID", values=runif(1)))

if (0) {
        options(width=120)
        plan <- mxComputeSequence(list(
            mxComputeOnce('fitfunction', 'fit'),
            mxComputeNumericDeriv(checkGradient=FALSE,
                                  hessian=FALSE, iterations=2),
            mxComputeReportDeriv(),
```

```
            mxComputeReportExpectation()
        ))

        sMod$expectation$.rampart <- 0L
        square <- mxRun(mxModel(sMod, plan))

        sMod$expectation$.rampart <- 2L
        rotated <- mxRun(mxModel(sMod, plan))

        ex <- square$expectation
        ex <- rotated$expectation
        eo <- ex$output
        ed <- ex$debug
        print(ed$layout)
        print(ed$rampartUsage)
        print(ed$numGroups)
        table(ed$layout$group)
        head(ed$layout[ed$layout$group == 1, ], n=20)
        #print(round(ed$A[1:20,1:20],2))
        #print(round(ed$rA[1:20,1:20],2))
        #print(ed$mean)

        #omxCheckCloseEnough(ed$rampartUsage, c(11064L, 317L, 198L, 2L), 1L)
        print(abs(rotated$output$fit - square$output$fit))
        print(max(abs(rotated$output$gradient - square$output$gradient)))
#       omxCheckCloseEnough(rotated$output$gradient,
#               square$output$gradient, 1e-4)
}

fit1 <- mxRun(sMod)
summary(fit1)

omxCheckCloseEnough(fit1$output$fit, 17212.46, .01)
omxCheckCloseEnough(max(abs(fit1$output$gradient)), 0, .01)
ed <- fit1$expectation$debug
omxCheckCloseEnough(ed$rampartUsage, c(902, 97, 21))
omxCheckCloseEnough(ed$numGroups, 8L)
omxCheckCloseEnough(
    sapply(unique(ed$layout$group),
           function(x) length(unique(ed$layout[ed$layout$group==x, 'copy']))),
    c(1L, 805L, 97L, 94L, 15L, 4L, 6L, 3L))

plan <- mxComputeSequence(list(
```

```
    mxComputeOnce('expectation', 'distribution', 'flat'),
    mxComputeReportExpectation()
))
slow <- sMod
slow$expectation$.rampart <- 0L
slowEx <- mxRun(mxModel(slow, plan))
ed <- slowEx$expectation$debug
omxCheckTrue(length(ed$rampartUsage)==0)
# each (entire) district is an independent unit
omxCheckCloseEnough(sapply(
    unique(ed$layout$group),
    function(x) length(unique(ed$layout[ed$layout$group==x, 'copy']))),
                    rep(1L,5))

if (0) { # this takes about 1.5 hours
        #options(width=120)
        plan <- mxComputeSequence(list(
            mxComputeOnce('fitfunction', 'fit'),
            mxComputeNumericDeriv(checkGradient=FALSE,
                                  iterations=2, verbose=2L),
            mxComputeReportDeriv(),
            mxComputeReportExpectation()
        ))

        slow <- omxSetParameters(sMod, labels=names(coef(fit1)),
                                 values=coef(fit1))
        slow$expectation$.rampart <- 0L
        slowFit <- mxRun(mxModel(slow, plan))

        omxCheckTrue(all(eigen(slowFit$output$hessian)$val > 0))
        omxCheckCloseEnough(slowFit$output$fit, fit1$output$fit, 65)
        omxCheckCloseEnough(max(abs(slowFit$output$gradient)), 0, 60)
        omxCheckCloseEnough(max(abs(slowFit$output$hessian %*%
                                       solve(fit1$output$hessian))), 0, 1.5)
}
```

Appendix I

rampart.R

```
library(OpenMx)
library(mvtnorm)

#set.seed(1)   # $\theta_1$
```

```
set.seed(3)    # $\theta_2$

numIndicators <- 5

numSchools <- 7
numTeachers <- 3
numStudents <- 5

genStructure <- function(upper, fanout, keyname) {
        lowerData <- NULL
        for (sx in 1:nrow(upper)) {
                extraFanout <- sample.int(fanout, 1)
                lowerData <- rbind(lowerData, data.frame(
                    upper=upper[sx,1], skill=rnorm(fanout + extraFanout,
                                                   mean=upper[sx, 'skill'])))
        }
        colnames(lowerData)[[1]] <- colnames(upper)[[1]]
        lowerData[[keyname]] <- 1:nrow(lowerData)
        lowerData <- lowerData[,c(3,1,2)]
        lowerData
}

dataEnv <- new.env()

assign("schoolData", data.frame(schoolID=1:numSchools,
                                skill=rnorm(numSchools)), envir=dataEnv)
assign("teacherData", genStructure(dataEnv$schoolData,
                                   numTeachers, 'teacherID'), envir=dataEnv)
assign("studentData", genStructure(dataEnv$teacherData,
                                   numStudents, 'studentID'), envir=dataEnv)

createIndicators <- function(latentSkill, indicatorMean, indicatorVariance) {
    if (missing(indicatorMean)) {
        indicatorMean <- runif(numIndicators,min=-1,max=1)
    }
    if (missing(indicatorVariance)) {
        indicatorVariance <- rlnorm(numIndicators) / 8
    }
    ind <- matrix(NA, length(latentSkill), length(indicatorVariance))
    for (ix in 1:length(latentSkill)) {
            ind[ix,] <- sapply(
                indicatorVariance,
                function(sd) rnorm(1, mean=latentSkill[ix], sd=sd))
```

```
    }
    ind <- t(t(ind) + indicatorMean)
    colnames(ind) <- paste0('i', 1:length(indicatorVariance))
    as.data.frame(ind)
}

for (tbl in paste0(c('school', 'teacher', 'student'), 'Data')) {
    dataEnv[[tbl]] <- cbind(dataEnv[[tbl]],
                            createIndicators(dataEnv[[tbl]]$skill))
}

dataEnv$studentData$i1[runif(nrow(dataEnv$studentData)) > .8] <- NA
#teacherData$i4[runif(nrow(teacherData)) > .8] <- NA

mkSingleFactor <- function(latent=c()) {
        mxModel('template', type='RAM',
                manifestVars = paste0('i', 1:numIndicators),
                latentVars = c("skill",latent),
                mxPath(from='skill', arrows=2, labels="Var",
                       values=rlnorm(1), lbound=.01),
                mxPath(from=paste0('i',1:numIndicators), arrows=2,
                       values=rlnorm(1), labels="Err", lbound=.01),
                mxPath(from="one", to=paste0('i',1:numIndicators),
                       free=TRUE, values=rnorm(4)),
                mxPath(from='skill', to=paste0('i',1:numIndicators),
                       labels=paste0('L',1:numIndicators), lbound=0,
                       values=c(1, runif(numIndicators-1, .5,1.5)),
                       free=c(FALSE, rep(TRUE,numIndicators-1)))
                )
}

singleFactor <- mkSingleFactor(NULL)

relabel <- function(m, prefix) {
  for (mat in c("A","S")) {
    lab <- m[[mat]]$labels
    lab[!is.na(lab)] <- paste0(prefix, lab[!is.na(lab)])
   m[[mat]]$labels <- lab
  }
  mxModel(m, name=prefix)
}

schMod <- mxModel(relabel(mkSingleFactor(), "school"),
```

```
                    mxData(type="raw", observed=dataEnv$schoolData,
                           primaryKey="schoolID", sort=FALSE))

tMod <- mxModel(relabel(singleFactor, "teacher"), schMod,
                    mxData(type="raw", observed=dataEnv$teacherData,
                           primaryKey="teacherID", sort=FALSE),
                    mxPath(from='school.skill', to='skill',
                           joinKey="schoolID", values=runif(1)))

sMod <- mxModel(relabel(singleFactor, "student"), tMod,
                  mxData(type="raw", observed=dataEnv$studentData,
                         primaryKey="studentID", sort=FALSE),
                    mxPath(from='teacher.skill', to='skill',
                           joinKey="teacherID", values=runif(1)))

interest <- c('wallTime', 'infoDefinite',
              'conditionNumber', 'fit', 'timestamp')

if (1) {
    result <- expand.grid(rampart=c(TRUE,FALSE), rep=1:200, gradient=NA)
    for (e1 in names(coef(sMod))) result[[e1]] <- NA
    for (i1 in interest) result[[i1]] <- NA
} else {
    load("/tmp/rampart.rda")
}

plan <- mxComputeSequence(list(
    mxComputeGradientDescent(),
    mxComputeNumericDeriv(iterations=2L),
    mxComputeHessianQuality(),
    mxComputeReportDeriv()
))

for (rrow in 1:nrow(result)) {
    if (!is.na(result[rrow, 'wallTime'])) next
#    if (!result[rrow, 'rampart']) next

    if (result[rrow, 'rampart']==FALSE &&
        !result[result$rep == result[rrow, 'rep'] &
                result$rampart==TRUE, 'infoDefinite']) {
        print("skip")
        next
    }
```

```

    set.seed(result[rrow, 'rep'])
    trial <- mxGenerateData(sMod, returnModel=TRUE)

    if (result[rrow, 'rampart']) {
        trial$expectation$.rampart <- as.integer(NA)
    } else {
        trial$expectation$.rampart <- 0L
        trial$fitfunction$parallel <- TRUE
    }
    trialFit <- mxRun(mxModel(trial, plan))

    result[rrow, names(coef(trialFit))] <- coef(trialFit)
    result[rrow, interest] <- trialFit$output[interest]
    result[rrow, 'gradient'] <- max(abs(trialFit$output$gradient))

    save(result, file="/tmp/rampart.rda")
}

sum(!is.na(result[result$rampart==TRUE, 'conditionNumber']))
sum(!is.na(result[result$rampart==FALSE, 'conditionNumber']))

cnMask <- (result$conditionNumber <
               median(result$conditionNumber, na.rm=TRUE) +
                   5 * mad(result$conditionNumber, na.rm=TRUE))
bothOkay <- cnMask[result$rampart==TRUE] & cnMask[result$rampart==FALSE]
length(which(bothOkay))

good <- result[result$rep %in% which(bothOkay),]
good[,c("rep",'rampart', "conditionNumber", 'gradient')]

cor(good[good$rampart==TRUE,"conditionNumber"],
    good[good$rampart==FALSE,"conditionNumber"])
cor(good[good$rampart==TRUE,"fit"],
    good[good$rampart==FALSE,"fit"])

summary <- c(rMean=norm(colMeans(good[good$rampart==TRUE,
                  names(coef(sMod))]) - coef(sMod), "2"),
             fMean=norm(colMeans(good[good$rampart==FALSE,
                  names(coef(sMod))]) - coef(sMod), "2"),
             rVar=norm(apply(good[good$rampart==TRUE,
                  names(coef(sMod))], 2, var), "2"),
             fVar=norm(apply(good[good$rampart==FALSE,
```

```
                    names(coef(sMod))], 2, var), "2"))
print(summary)

if (0) {
        library(ggplot2)
        ggplot(good) + geom_histogram(aes(wallTime)) +
            facet_wrap(~rampart, scales="free_x")
}
```

Appendix J

boker2009Compare.R

```
library(nlme)
library(OpenMx)
options(width=120)
mxOption(NULL, 'Optimality␣tolerance', "1e-13")

load("e2Pairing.rda")
load("tFrame.rda")

if (1) {
    # otherwise OpenMx has trouble finding the same mode as nlme
    for (f in c('selfyRotFV', 'otheryRotFV', 'selfxRotFV', 'otherxRotFV')) {
        tFrame[[f]] <- log(1+tFrame[[f]])
    }
}

# ------------------------------ original analysis

# table 1: head anterior-posterior RMS angular velocity
headAPlme <- lme(selfyRotFV ~ selfSex + otherSex + isConfed +
                    dampHead + dampFace + dampVoice +
                    otheryRotFV + confedByOtherSex + confedByDampHead +
                    confedByDampFace + confedByDampVoice,
            random= ~ 1 | naiveID, data=tFrame, method="ML")

# table 2: head lateral RMS angular velocity
headLlme <- lme(selfxRotFV ~ selfSex + otherSex + isConfed +
                    dampHead + dampFace + dampVoice +
                    otherxRotFV + confedByOtherSex + confedByDampHead +
                    confedByDampFace + confedByDampVoice,
            random= ~ 1 | naiveID, data=tFrame, method="ML")

```

```
# ---------------------------------------- rSEM

for (col in c('naiveID', 'confedID')) {
    e2Pairing[[col]] <- as.integer(e2Pairing[[col]])
}

pairHash <- e2Pairing$confedID * 100L + e2Pairing$naiveID
pairData <- e2Pairing[!duplicated(pairHash),
                      c('naiveID', 'confedID', 'naiveSex', 'confedSex')]
pairData <- cbind(pairID=pairData$confedID * 100L +
                      pairData$naiveID, pairData)
pairData$oppositeSex <-
    as.numeric(pairData[, 'naiveSex'] != pairData[, 'confedSex'])

response <- c("selfxRotFV", "selfyRotFV")
zeroVarPred <- c(paste0('damp', c('Head','Face','Voice')),
                 paste0(c('self', 'other'), 'Sex'), 'isConfed',
                 "confedByOtherSex", "confedByDampHead",
                 "confedByDampFace", "confedByDampVoice")

tFrame$pairID <- as.integer(tFrame$confedID * 100L + tFrame$naiveID)

naiveIndModel <- mxModel(
    model="naive", type="RAM",
    latentVars=c('xIntercept', 'yIntercept'),
    mxData(e2Pairing[!duplicated(e2Pairing$naiveID),
                     c('naiveID'), drop=FALSE],
           type="raw", primaryKey="naiveID"),
    mxPath('xIntercept', arrows=2, values=1,
           lbound=1e-3, labels="naiveVaryInt_x"),
    mxPath('yIntercept', arrows=2, values=1,
           lbound=1e-3, labels="naiveVaryInt_y"))

confedEmptyModel <- mxModel(
    model="confed", type="RAM",
    latentVars=c('xIntercept', 'yIntercept'),
    mxData(e2Pairing[!duplicated(e2Pairing$confedID),
                     c('confedID'), drop=FALSE],
           type="raw", primaryKey="confedID"),
    mxPath('xIntercept', arrows=2, values=0, free=FALSE, lbound=1e-3),
    mxPath('yIntercept', arrows=2, values=0, free=FALSE, lbound=1e-3))

pairModelOrig <- mxModel(
```

```
    model="pair", type="RAM", naiveIndModel, confedEmptyModel,
    latentVars=c('naiveXIntercept', 'confedXIntercept',
        'naiveYIntercept', 'confedYIntercept',
        'oppositeSex'),
    mxData(pairData, type="raw", primaryKey="pairID"),
    mxPath('one', 'oppositeSex', free=FALSE, labels="data.oppositeSex"),
    mxPath('naive.xIntercept', 'naiveXIntercept',
           free=FALSE, values=1, joinKey="naiveID"),
    mxPath('naive.yIntercept', 'naiveYIntercept',
           free=FALSE, values=1, joinKey="naiveID"),
    mxPath('confed.xIntercept', 'confedXIntercept',
           free=FALSE, values=1, joinKey="confedID"),
    mxPath('confed.yIntercept', 'confedYIntercept',
           free=FALSE, values=1, joinKey="confedID"))

oneMinuteOrig <- mxModel(
    model="original", type="RAM", pairModelOrig,
    manifestVars=response,
    latentVars=c(zeroVarPred, "otheryRotFV", "otherxRotFV"),
    mxData(tFrame, type="raw", sort=FALSE),
    mxPath('one', zeroVarPred, free=FALSE,
           labels=paste0('data.', zeroVarPred)),
    mxPath('one', c("otheryRotFV", "otherxRotFV"), free=FALSE,
           labels=paste0('data.', c("otheryRotFV", "otherxRotFV"))),
    mxPath('pair.naiveXIntercept', 'selfxRotFV', free=FALSE,
           values=1, joinKey="pairID"),
    mxPath('pair.naiveYIntercept', 'selfyRotFV', free=FALSE,
           values=1, joinKey="pairID"),
    mxPath(response, arrows=2, connect="single"),
    mxPath('one', response, labels=paste0(response, "_int")),
    mxPath('otherxRotFV', 'selfxRotFV', labels="otherxRotFV_on_x"),
    mxPath('otheryRotFV', 'selfyRotFV', labels="otheryRotFV_on_y"),
    mxPath(zeroVarPred, c("selfyRotFV"), connect="all.pairs",
           labels=paste0(zeroVarPred, "_on_y")),
    mxPath(zeroVarPred, c("selfxRotFV"), connect="all.pairs",
           labels=paste0(zeroVarPred, "_on_x")))

oneMinuteOrig$S$values[response,response] <- diag(length(response))
oneMinuteOrig$expectation$.ignoreDefVarsHack <- TRUE

oneMinuteOrigFit <- mxRun(oneMinuteOrig) #, checkpoint=TRUE)
#summary(oneMinuteOrigFit)

```

```
omxCheckCloseEnough(logLik(oneMinuteOrigFit),
                    -1207.711, 1e-2)
omxCheckCloseEnough(logLik(oneMinuteOrigFit) -
                        (logLik(headLlme) + logLik(headAPlme)), 0, 1e-6)

# -------------------------- comparison models

# covariance between x & y but no varying intercept for naive

oneMinuteV2 <- mxModel(
    model="xyCov", type="RAM", pairModelOrig,
    manifestVars=response,
    latentVars=c(zeroVarPred, "otheryRotFV", "otherxRotFV"),
    mxData(tFrame, type="raw", sort=FALSE),
    mxPath('one', zeroVarPred, free=FALSE,
           labels=paste0('data.', zeroVarPred)),
    mxPath('one', c("otheryRotFV", "otherxRotFV"), free=FALSE,
           labels=paste0('data.', c("otheryRotFV", "otherxRotFV"))),
    mxPath('pair.naiveXIntercept', 'selfxRotFV', free=FALSE,
           values=1, joinKey="pairID"),
    mxPath('pair.naiveYIntercept', 'selfyRotFV', free=FALSE,
           values=1, joinKey="pairID"),
    mxPath(response, arrows=2, connect="unique.pairs"),
    mxPath('one', response, labels=paste0(response, "_int")),
    mxPath('otherxRotFV', 'selfxRotFV', labels="otherxRotFV_on_x"),
    mxPath('otheryRotFV', 'selfyRotFV', labels="otheryRotFV_on_y"),
    mxPath(zeroVarPred, c("selfyRotFV"), connect="all.pairs",
           labels=paste0(zeroVarPred, "_on_y")),
    mxPath(zeroVarPred, c("selfxRotFV"), connect="all.pairs",
           labels=paste0(zeroVarPred, "_on_x")))

oneMinuteV2$S$values[response,response] <- diag(length(response))
oneMinuteV2$S$labels[1,2] <- 'xyCov'
oneMinuteV2$S$labels[2,1] <- 'xyCov'
oneMinuteV2$expectation$.ignoreDefVarsHack <- TRUE
oneMinuteV2Fit <- mxRun(oneMinuteV2) #, checkpoint=TRUE)

naiveModel <- mxModel(
    model="naive", type="RAM",
    latentVars=c('xIntercept', 'yIntercept'),
    mxData(e2Pairing[!duplicated(e2Pairing$naiveID),
                     c('naiveID'), drop=FALSE],
           type="raw", primaryKey="naiveID"),
```

```
    mxPath('xIntercept', arrows=2, values=1,
           lbound=1e-3, labels="naiveVaryInt_x"),
    mxPath('yIntercept', arrows=2, values=1,
           lbound=1e-3, labels="naiveVaryInt_y"))

confedModel <- mxModel(
    model="confed", type="RAM",
    latentVars=c('xIntercept', 'yIntercept'),
    mxData(e2Pairing[!duplicated(e2Pairing$confedID),
                     c('confedID'), drop=FALSE],
           type="raw", primaryKey="confedID"),
    mxPath('xIntercept', arrows=2, values=1,
           lbound=1e-3, labels="confedVaryInt_x"),
    mxPath('yIntercept', arrows=2, values=1,
           lbound=1e-3, labels="confedVaryInt_y"))

pairModel <- mxModel(
    model="pair", type="RAM", naiveModel, confedModel,
    latentVars=c('naiveXIntercept', 'confedXIntercept',
        'naiveYIntercept', 'confedYIntercept',
        'oppositeSex'),
    mxData(pairData, type="raw", primaryKey="pairID"),
    mxPath('one', 'oppositeSex', free=FALSE, labels="data.oppositeSex"),
    mxPath('naive.xIntercept', 'naiveXIntercept',
           free=FALSE, values=1, joinKey="naiveID"),
    mxPath('naive.yIntercept', 'naiveYIntercept',
           free=FALSE, values=1, joinKey="naiveID"),
    mxPath('confed.xIntercept', 'confedXIntercept',
           free=FALSE, values=1, joinKey="confedID"),
    mxPath('confed.yIntercept', 'confedYIntercept',
           free=FALSE, values=1, joinKey="confedID"))

# naive & confed varying intercept and covariance between x & y

oneMinuteV1 <- mxModel(
    model="xyCov_confed", type="RAM", pairModel,
    manifestVars=response,
    latentVars=c(zeroVarPred, "otheryRotFV", "otherxRotFV"),
    mxData(tFrame, type="raw", sort=FALSE),
    mxPath('one', zeroVarPred, free=FALSE,
           labels=paste0('data.', zeroVarPred)),
    mxPath('one', c("otheryRotFV", "otherxRotFV"), free=FALSE,
           labels=paste0('data.', c("otheryRotFV", "otherxRotFV"))),
```

```
    mxPath('pair.confedXIntercept', 'selfxRotFV', free=FALSE,
           values=1, joinKey="pairID"),
    mxPath('pair.confedYIntercept', 'selfyRotFV', free=FALSE,
           values=1, joinKey="pairID"),
    mxPath('pair.naiveXIntercept', 'selfxRotFV', free=FALSE,
           values=1, joinKey="pairID"),
    mxPath('pair.naiveYIntercept', 'selfyRotFV', free=FALSE,
           values=1, joinKey="pairID"),
    mxPath(response, arrows=2, connect="unique.pairs"),
    mxPath('one', response, labels=paste0(response, "_int")),
    mxPath('otherxRotFV', 'selfxRotFV', labels="otherxRotFV_on_x"),
    mxPath('otheryRotFV', 'selfyRotFV', labels="otheryRotFV_on_y"),
    mxPath(zeroVarPred, c("selfyRotFV"), connect="all.pairs",
           labels=paste0(zeroVarPred, "_on_y")),
    mxPath(zeroVarPred, c("selfxRotFV"), connect="all.pairs",
           labels=paste0(zeroVarPred, "_on_x")))

oneMinuteV1$S$values[response,response] <- diag(length(response))
oneMinuteV1$S$labels[1,2] <- 'xyCov'
oneMinuteV1$S$labels[2,1] <- 'xyCov'
oneMinuteV1$expectation$.ignoreDefVarsHack <- TRUE
oneMinuteV1Fit <- mxRun(oneMinuteV1) #, checkpoint=TRUE)

# naive & confed varying intercept but no covariance between x & y

oneMinuteV3 <- mxModel(
    model="only_confed", type="RAM", pairModel,
    manifestVars=response,
    latentVars=c(zeroVarPred, "otheryRotFV", "otherxRotFV"),
    mxData(tFrame, type="raw", sort=FALSE),
    mxPath('one', zeroVarPred, free=FALSE,
           labels=paste0('data.', zeroVarPred)),
    mxPath('one', c("otheryRotFV", "otherxRotFV"), free=FALSE,
           labels=paste0('data.', c("otheryRotFV", "otherxRotFV"))),
    mxPath('pair.confedXIntercept', 'selfxRotFV', free=FALSE,
           values=1, joinKey="pairID"),
    mxPath('pair.confedYIntercept', 'selfyRotFV', free=FALSE,
           values=1, joinKey="pairID"),
    mxPath('pair.naiveXIntercept', 'selfxRotFV', free=FALSE,
           values=1, joinKey="pairID"),
    mxPath('pair.naiveYIntercept', 'selfyRotFV', free=FALSE,
           values=1, joinKey="pairID"),
    mxPath(response, arrows=2, connect="single"),
```

```
    mxPath('one', response, labels=paste0(response, "_int")),
    mxPath('otherxRotFV', 'selfxRotFV', labels="otherxRotFV_on_x"),
    mxPath('otheryRotFV', 'selfyRotFV', labels="otheryRotFV_on_y"),
    mxPath(zeroVarPred, c("selfyRotFV"), connect="all.pairs",
           labels=paste0(zeroVarPred, "_on_y")),
    mxPath(zeroVarPred, c("selfxRotFV"), connect="all.pairs",
           labels=paste0(zeroVarPred, "_on_x")))

oneMinuteV3$S$values[response,response] <- diag(length(response))
oneMinuteV3$expectation$.ignoreDefVarsHack <- TRUE
oneMinuteV3Fit <- mxRun(oneMinuteV3) #, checkpoint=TRUE)

# add covariance for varying intercepts

naiveCModel <- mxModel(
    model="naive", type="RAM",
    latentVars=c('xIntercept', 'yIntercept'),
    mxData(e2Pairing[!duplicated(e2Pairing$naiveID),
                     c('naiveID'), drop=FALSE],
           type="raw", primaryKey="naiveID"),
    mxPath('xIntercept', arrows=2, values=1,
           lbound=1e-3, labels="naiveVaryInt_x"),
    mxPath('xIntercept', 'yIntercept', arrows=2,
           labels="naiveVaryInt_cov"),
    mxPath('yIntercept', arrows=2, values=1,
           lbound=1e-3, labels="naiveVaryInt_y"))

confedCModel <- mxModel(
    model="confed", type="RAM",
    latentVars=c('xIntercept', 'yIntercept'),
    mxData(e2Pairing[!duplicated(e2Pairing$confedID),
                     c('confedID'), drop=FALSE],
           type="raw", primaryKey="confedID"),
    mxPath('xIntercept', arrows=2, values=1,
           lbound=1e-3, labels="confedVaryInt_x"),
    mxPath('xIntercept', 'yIntercept', arrows=2,
           labels="confedVaryInt_cov"),
    mxPath('yIntercept', arrows=2, values=1,
           lbound=1e-3, labels="confedVaryInt_y"))

pairCModel <- mxModel(
    model="pair", type="RAM", naiveCModel, confedCModel,
    latentVars=c('naiveXIntercept', 'confedXIntercept',
```

```
        'naiveYIntercept', 'confedYIntercept',
        'oppositeSex'),
    mxData(pairData, type="raw", primaryKey="pairID"),
    mxPath('one', 'oppositeSex', free=FALSE, labels="data.oppositeSex"),
    mxPath('naive.xIntercept', 'naiveXIntercept',
           free=FALSE, values=1, joinKey="naiveID"),
    mxPath('naive.yIntercept', 'naiveYIntercept',
           free=FALSE, values=1, joinKey="naiveID"),
    mxPath('confed.xIntercept', 'confedXIntercept',
           free=FALSE, values=1, joinKey="confedID"),
    mxPath('confed.yIntercept', 'confedYIntercept',
           free=FALSE, values=1, joinKey="confedID"))

oneMinuteV4 <- mxModel(
    model="full", type="RAM", pairCModel,
    manifestVars=response,
    latentVars=c(zeroVarPred, "otheryRotFV", "otherxRotFV"),
    mxData(tFrame, type="raw", sort=FALSE),
    mxPath('one', zeroVarPred, free=FALSE,
           labels=paste0('data.', zeroVarPred)),
    mxPath('one', c("otheryRotFV", "otherxRotFV"), free=FALSE,
           labels=paste0('data.', c("otheryRotFV", "otherxRotFV"))),
    mxPath('pair.confedXIntercept', 'selfxRotFV', free=FALSE,
           values=1, joinKey="pairID"),
    mxPath('pair.confedYIntercept', 'selfyRotFV', free=FALSE,
           values=1, joinKey="pairID"),
    mxPath('pair.naiveXIntercept', 'selfxRotFV', free=FALSE,
           values=1, joinKey="pairID"),
    mxPath('pair.naiveYIntercept', 'selfyRotFV', free=FALSE,
           values=1, joinKey="pairID"),
    mxPath(response, arrows=2, connect="unique.pairs"),
    mxPath('one', response, labels=paste0(response, "_int")),
    mxPath('otherxRotFV', 'selfxRotFV', labels="otherxRotFV_on_x"),
    mxPath('otheryRotFV', 'selfyRotFV', labels="otheryRotFV_on_y"),
    mxPath(zeroVarPred, c("selfyRotFV"), connect="all.pairs",
           labels=paste0(zeroVarPred, "_on_y")),
    mxPath(zeroVarPred, c("selfxRotFV"), connect="all.pairs",
           labels=paste0(zeroVarPred, "_on_x")))

oneMinuteV4$S$values[response,response] <- diag(length(response))
oneMinuteV4$S$labels[1,2] <- 'xyCov'
oneMinuteV4$S$labels[2,1] <- 'xyCov'
oneMinuteV4$expectation$.ignoreDefVarsHack <- TRUE
```

```r
oneMinuteV4Fit <- mxRun(oneMinuteV4) #, checkpoint=TRUE)

save(oneMinuteV4Fit, oneMinuteV1Fit, oneMinuteV2Fit,
     oneMinuteV3Fit, oneMinuteOrigFit, file="boker2009Compare.rda")

mxCompare(oneMinuteV4Fit, list(oneMinuteV1Fit, oneMinuteV2Fit,
                               oneMinuteV3Fit, oneMinuteOrigFit))
```

Appendix K

boker2009Sim.R

```r
library(OpenMx)
options(width=120)
mxOption(NULL, 'Optimality tolerance', "1e-13")

load("e2Pairing.rda")
load("tFrame.rda")

if (1) {
    for (f in c('selfyRotFV', 'otheryRotFV',
                'selfxRotFV', 'otherxRotFV')) {
        tFrame[[f]] <- log(1+tFrame[[f]])
    }
}

for (col in c('naiveID', 'confedID')) {
    e2Pairing[[col]] <- as.integer(e2Pairing[[col]])
}

pairHash <- e2Pairing$confedID * 100L + e2Pairing$naiveID
pairData <- e2Pairing[!duplicated(pairHash),
                      c('naiveID', 'confedID', 'naiveSex', 'confedSex')]
pairData <- cbind(pairID=pairData$confedID * 100L +
                      pairData$naiveID, pairData)
pairData$oppositeSex <-
    as.numeric(pairData[, 'naiveSex'] != pairData[, 'confedSex'])

response <- c("selfxRotFV", "selfyRotFV")
zeroVarPred <- c(paste0('damp', c('Head','Face','Voice')),
                 paste0(c('self', 'other'), 'Sex'), 'isConfed',
                 "confedByOtherSex", "confedByDampHead",
                 "confedByDampFace", "confedByDampVoice")

```

```
tFrame$pairID <- as.integer(tFrame$confedID * 100L + tFrame$naiveID)

naiveIndModel <- mxModel(
    model="naive", type="RAM",
    latentVars=c('xIntercept', 'yIntercept'),
    mxData(e2Pairing[!duplicated(e2Pairing$naiveID),
                     c('naiveID'), drop=FALSE],
           type="raw", primaryKey="naiveID"),
    mxPath('xIntercept', arrows=2, values=1,
           lbound=1e-3, labels="naiveVaryInt_x"),
    mxPath('yIntercept', arrows=2, values=1,
           lbound=1e-3, labels="naiveVaryInt_y"))

confedEmptyModel <- mxModel(
    model="confed", type="RAM",
    latentVars=c('xIntercept', 'yIntercept'),
    mxData(e2Pairing[!duplicated(e2Pairing$confedID),
                     c('confedID'), drop=FALSE],
           type="raw", primaryKey="confedID"),
    mxPath('xIntercept', arrows=2, values=0, free=FALSE, lbound=1e-3),
    mxPath('yIntercept', arrows=2, values=0, free=FALSE, lbound=1e-3))

pairModelOrig <- mxModel(
    model="pair", type="RAM", naiveIndModel, confedEmptyModel,
    latentVars=c('naiveXIntercept', 'confedXIntercept',
        'naiveYIntercept', 'confedYIntercept',
        'oppositeSex'),
    mxData(pairData, type="raw", primaryKey="pairID"),
    mxPath('one', 'oppositeSex', free=FALSE, labels="data.oppositeSex"),
    mxPath('naive.xIntercept', 'naiveXIntercept',
           free=FALSE, values=1, joinKey="naiveID"),
    mxPath('naive.yIntercept', 'naiveYIntercept',
           free=FALSE, values=1, joinKey="naiveID"),
    mxPath('confed.xIntercept', 'confedXIntercept',
           free=FALSE, values=1, joinKey="confedID"),
    mxPath('confed.yIntercept', 'confedYIntercept',
           free=FALSE, values=1, joinKey="confedID"))

oneMinuteOrig <- mxModel(
    model="original", type="RAM", pairModelOrig,
    manifestVars=response,
    latentVars=c(zeroVarPred, "otheryRotFV", "otherxRotFV"),
    mxData(tFrame, type="raw", sort=FALSE),
```

```
    mxPath('one', zeroVarPred, free=FALSE,
           labels=paste0('data.', zeroVarPred)),
    mxPath('one', c("otheryRotFV", "otherxRotFV"), free=FALSE,
           labels=paste0('data.', c("otheryRotFV", "otherxRotFV"))),
    mxPath('pair.naiveXIntercept', 'selfxRotFV', free=FALSE,
           values=1, joinKey="pairID"),
    mxPath('pair.naiveYIntercept', 'selfyRotFV', free=FALSE,
           values=1, joinKey="pairID"),
    mxPath(response, arrows=2, connect="single"),
    mxPath('one', response, labels=paste0(response, "_int")),
    mxPath('otherxRotFV', 'selfxRotFV', labels="otherxRotFV_on_x"),
    mxPath('otheryRotFV', 'selfyRotFV', labels="otheryRotFV_on_y"),
    mxPath(zeroVarPred, c("selfyRotFV"), connect="all.pairs",
           labels=paste0(zeroVarPred, "_on_y")),
    mxPath(zeroVarPred, c("selfxRotFV"), connect="all.pairs",
           labels=paste0(zeroVarPred, "_on_x")))

oneMinuteOrig$S$values[response,response] <- diag(length(response))
oneMinuteOrig$expectation$.ignoreDefVarsHack <- TRUE

oneMinuteOrigFit <- mxRun(oneMinuteOrig) #, checkpoint=TRUE)

# ------------------------------ comparison model

naiveModel <- mxModel(
    model="naive", type="RAM",
    latentVars=c('xIntercept', 'yIntercept'),
    mxData(e2Pairing[!duplicated(e2Pairing$naiveID),
                     c('naiveID'), drop=FALSE],
           type="raw", primaryKey="naiveID"),
    mxPath('xIntercept', arrows=2, values=1,
           lbound=1e-3, labels="naiveVaryInt_x"),
    mxPath('yIntercept', arrows=2, values=1,
           lbound=1e-3, labels="naiveVaryInt_y"))

confedModel <- mxModel(
    model="confed", type="RAM",
    latentVars=c('xIntercept', 'yIntercept'),
    mxData(e2Pairing[!duplicated(e2Pairing$confedID),
                     c('confedID'), drop=FALSE],
           type="raw", primaryKey="confedID"),
    mxPath('xIntercept', arrows=2, values=1,
           lbound=1e-3, labels="confedVaryInt_x"),
```

```
    mxPath('yIntercept', arrows=2, values=1,
           lbound=1e-3, labels="confedVaryInt_y"))

pairModel <- mxModel(
    model="pair", type="RAM", naiveModel, confedModel,
    latentVars=c('naiveXIntercept', 'confedXIntercept',
        'naiveYIntercept', 'confedYIntercept',
        'oppositeSex'),
    mxData(pairData, type="raw", primaryKey="pairID"),
    mxPath('one', 'oppositeSex', free=FALSE, labels="data.oppositeSex"),
    mxPath('naive.xIntercept', 'naiveXIntercept',
           free=FALSE, values=1, joinKey="naiveID"),
    mxPath('naive.yIntercept', 'naiveYIntercept',
           free=FALSE, values=1, joinKey="naiveID"),
    mxPath('confed.xIntercept', 'confedXIntercept',
           free=FALSE, values=1, joinKey="confedID"),
    mxPath('confed.yIntercept', 'confedYIntercept',
           free=FALSE, values=1, joinKey="confedID"))

# naive & confed varying intercept and covariance between x & y

oneMinuteSat <- mxModel(
    model="oneMinute", type="RAM", pairModel,
    manifestVars=response,
    latentVars=c(zeroVarPred, "otheryRotFV", "otherxRotFV"),
    mxData(tFrame, type="raw", sort=FALSE),
    mxPath('one', zeroVarPred, free=FALSE,
           labels=paste0('data.', zeroVarPred)),
    mxPath('one', c("otheryRotFV", "otherxRotFV"), free=FALSE,
           labels=paste0('data.', c("otheryRotFV", "otherxRotFV"))),
    mxPath('pair.confedXIntercept', 'selfxRotFV', free=FALSE,
           values=1, joinKey="pairID"),
    mxPath('pair.confedYIntercept', 'selfyRotFV', free=FALSE,
           values=1, joinKey="pairID"),
    mxPath('pair.naiveXIntercept', 'selfxRotFV', free=FALSE,
           values=1, joinKey="pairID"),
    mxPath('pair.naiveYIntercept', 'selfyRotFV', free=FALSE,
           values=1, joinKey="pairID"),
    mxPath(response, arrows=2, connect="unique.pairs"),
    mxPath('one', response, labels=paste0(response, "_int")),
    mxPath('otherxRotFV', 'selfxRotFV', labels="otherxRotFV_on_x"),
    mxPath('otheryRotFV', 'selfyRotFV', labels="otheryRotFV_on_y"),
    mxPath(zeroVarPred, c("selfyRotFV"), connect="all.pairs",
```

```
            labels=paste0(zeroVarPred, "_on_y")),
      mxPath(zeroVarPred, c("selfxRotFV"), connect="all.pairs",
            labels=paste0(zeroVarPred, "_on_x")))

oneMinuteSat$S$values[response,response] <- diag(length(response))
oneMinuteSat$S$labels[1,2] <- 'xyCov'
oneMinuteSat$S$labels[2,1] <- 'xyCov'
oneMinuteSat$expectation$.ignoreDefVarsHack <- TRUE
oneMinuteSatFit <- mxRun(oneMinuteSat) #, checkpoint=TRUE)

# ------------------------------------------ simulation

set.seed(1)
zScore <- oneMinuteSatFit$output$estimate /
    oneMinuteSatFit$output$standardErrors

candidate <- matrix(NA, ncol=length(zScore), nrow=5,
                    dimnames=list(c('absent', 'small+', 'small-',
                        'large+', 'large-'),
                        names(coef(oneMinuteSatFit))))

# don't care about means
for (par in paste0('self', c('x','y'), 'RotFV_int')) {
    candidate[,par] <- coef(oneMinuteSatFit)[par]
}

# don't care about variances
for (par in 1:2) {
    pname <- paste0('oneMinute.S[',par,',',par,']')
    candidate[,pname] <- coef(oneMinuteSatFit)[pname]
}

isLarge <- abs(zScore) > 2

for (p1 in c('naive','confed')) {
    for (p2 in c('VaryInt_x', 'VaryInt_y')) {
        par <- paste0(p1, p2)
        if (isLarge[par,]) {
            small <- 1.5 * oneMinuteSatFit$output$standardErrors[par,1]
            large <- coef(oneMinuteSatFit)[par]
        } else {
            small <- coef(oneMinuteSatFit)[par]
            large <- 3 * oneMinuteSatFit$output$standardErrors[par,1]
```

```
        }
        candidate[c('absent', 'small-', 'small+'), par] <- small
        candidate[c('large-', 'large+'), par] <- large
    }
}

for (par in names(coef(oneMinuteSatFit))[is.na(candidate['absent',])]) {
    if (isLarge[par,]) {
        large <- abs(coef(oneMinuteSatFit)[par])
        small <- 1.5 * oneMinuteSatFit$output$standardErrors[par,1]
    } else {
        large <- 3 * oneMinuteSatFit$output$standardErrors[par,1]
        small <- abs(coef(oneMinuteSatFit)[par])
    }
    candidate['large+',par] <- large
    candidate['large-',par] <- -large
    candidate['small+',par] <- small
    candidate['small-',par] <- -small
    candidate['absent',par] <- 0
}

paramOfInterest <- candidate['small+',] != candidate['large+',]

save(candidate, paramOfInterest, file="boker2009-sim.rda")

require("pROC")

startSeed <- 1
rda <- "/tmp/oneMinuteSim.rda"
if (1) {
    result <- NULL
} else {
    load(rda)
    startSeed <- 1L + max(result$seed)
}

for (rep in startSeed:100) {
    print(rep)
    set.seed(rep)
    s1 <- sample.int(5, ncol(candidate), replace=TRUE)
    parVec <- candidate[matrix(1:5, nrow=5, ncol=ncol(candidate)) ==
                              matrix(s1, byrow=TRUE, nrow=5,
                                     ncol=ncol(candidate))]
```

```
    names(parVec) <- colnames(candidate)

    simModel1 <- mxGenerateData(omxSetParameters(
        oneMinuteSat, labels=names(coef(oneMinuteSatFit)), values=parVec),
                                returnModel=TRUE)

    simFit1 <- mxRun(simModel1, checkpoint=TRUE)

    simModel2 <- omxSetParameters(
        oneMinuteOrig, labels=names(coef(oneMinuteSatFit)),
        values=parVec, strict=FALSE)
    simModel2$data$observed <- simModel1$data$observed
    simFit2 <- mxRun(simModel2, checkpoint=TRUE)

    # could fit them as a group of independent models TODO

    fits <- list("sat"=simFit1, "orig"=simFit2)
    for (mx in 1:2) {
        fit <- fits[[mx]]
        evidence <- (fit$output$estimate / fit$output$standardErrors)[,]
        if (fit$output$status$code != 0 || any(is.na(evidence))) {
            cat(paste(names(fits)[mx], rep, "got status",
                      fit$output$status$code), fill=TRUE)
            next
        }
        evidence <- evidence[ names(evidence) %in%
                              colnames(candidate)[paramOfInterest] ]
        mask <- match(names(evidence), names(parVec))
        wrongSign <- c(sign(evidence)) != sign(parVec[mask]) & (s1[mask] >= 4)

        df <- data.frame(model=names(fits)[mx],
                         seed=rep,
                         found=(ifelse(wrongSign, -1.0, 1.0) * abs(evidence)),
                         effect=(s1 >= 4)[mask])
        result <- rbind(result, df)
    }

    save(result, file=rda)

    pdf(file="roc.pdf")
    roc(effect ~ found, result[result$model=="orig",],
        plot=T, col="red")
    roc(effect ~ found, result[result$model=="sat",],
```

```
            plot=T, col="green", add=TRUE)
    dev.off()
}
```