Behavior Research Methods (2021) 53:2127–2157
https://doi.org/10.3758/s13428-020-01532-y

R-squared change in structural equation models with latent variables and missing data

Timothy Hayes[1]

Accepted: 20 December 2020 / Published online: 29 March 2021

Abstract

Researchers frequently wish to make incremental validity claims, suggesting that a construct of interest significantly predicts a given outcome when controlling for other overlapping constructs and potential confounders. Once the significance of such an effect has been established, it is good practice to also assess and report its magnitude. In OLS regression, this is easily accomplished by calculating the change in *R*-squared, ΔR^2, between one's full model and a reduced model that omits all but the target predictor(s) of interest. Because observed variable regression methods ignore measurement error, however, their estimates are prone to bias and inflated type I error rates. As a result, researchers are increasingly encouraged to switch from observed variable modeling conducted in the regression framework to latent variable modeling conducted in the structural equation modeling (SEM) framework. Standard SEM software packages provide overall R^2 measures for each outcome, yet calculation of ΔR^2 is not intuitive in models with latent variables. Omitting all indicators of a latent factor in a reduced model will alter the overidentifying constraints imposed on the model, affecting parameter estimation and fit. Furthermore, omitting variables in a reduced model may affect estimation under missing data, particularly when conditioning on those variables is essential to meeting the MAR assumption. In this article, I describe four approaches to calculating ΔR^2 in SEMs with latent variables and missing data, compare their performance via simulation, describe a set of extensions to the methods, and provide a set of R functions for calculating ΔR^2 in SEM.

Keywords *R*-squared · *R*-squared change · Effect size · Incremental validity · Structural equation modeling (SEM) · Missing data

Researchers frequently wish to argue that a construct of theoretical interest significantly influences an outcome variable in a manner that is distinct and separable from the influences of other overlapping constructs and confounders (Hunsley & Meyer, 2003; Sechrest, 1963; Wang & Eastwick, 2020; Westfall & Yarkoni, 2016). When the effect associated with a key predictor variable remains significant in an analysis that includes all relevant control variables, the predictor is deemed to have *incremental validity*, providing unique prediction of the outcome variable even after controlling for the set of competing predictors. Yet, even if the effect of a key predictor variable on a given outcome remains *statistically significant* when controlling for a set of covariates, the variable will be of limited utility if its effect is too small to be considered *practically significant* (Kirk, 1996). For this reason, researchers are well advised to compute and report a measure of effect size along with their analysis.

Within the regression framework, effect size is typically based on the proportion of variance explained in one's outcome by a set of predictor variables – that is, multiple *R*-squared (and the related f^2; see Cohen, 1988). As the preceding discussion implies, although researchers may at times be interested in the proportion of variance explained by the set of all predictors in a given model (the model R^2), they are quite often interested in the increment in variance uniquely explained by a key predictor variable or set of predictor variables. This increment in R^2 (ΔR^2) is typically computed as the difference between an R^2 obtained from one's full model, R^2_{Full} and an R^2 obtained from a reduced model that omits the target predictors of interest, i.e.: $R^2_{Reduced}$ – i.e., *R-squared Change*, calculated as:

$$\Delta R^2 = R^2_{Full} - R^2_{Reduced}. \tag{1}$$

These procedures for incremental validity testing and effect size estimation are easily implemented in the ordinary least squares (OLS) regression framework, yet the accuracy of these manifest variable regression methods is undermined by their reliance on the unrealistic assumption that all predictor

✉ Timothy Hayes
thayes@fiu.edu

[1] Department of Psychology, Florida International University, 11200 SW 8 Street, DM 381B, Miami, FL 33199, USA

variables are measured without error (Cohen et al., 2003; Pedhazur, 1997). Although the harmful effects of ignoring measurement error in one's analysis have long been known (see e.g., Bollen, 1989, ch. 5) the use of observed variable methods has remained widespread, leading to a recent groundswell of renewed interest in demonstrating to a modern audience the myriad ways in which ignoring measurement error in one's analyses can lead to biased parameter estimates and inflated type I error rates (Cole & Preacher, 2014; Ledgerwood & Shrout, 2011; Savalei, 2018; Westfall & Yarkoni, 2016). The fact that parameter estimates from observed variable regression analyses are prone to becoming distorted and untrustworthy has led to increasingly forceful recommendations from methodological experts urging researchers to switch from manifest variable models implemented in the OLS regression framework to latent variable models implemented in the structural equation modeling (SEM) framework when testing incremental validity claims (Cole & Preacher, 2014; Wang & Eastwick, 2020; Westfall & Yarkoni, 2016).

With respect to using significance testing to support an incremental validity claim, the transition from OLS to SEM is straightforward: simply assess whether the (partial) structural regression coefficient associated with one's target latent variable of interest remains significant when controlling for the influences of the other latent covariates included in the model. With respect to computing the effect size associated with an incremental validity claim, at first glance, it seems logical to assume that computing ΔR^2 in latent variable models would be similarly straightforward. After all, most widely used SEM software packages can output the overall model R^2 for each endogenous outcome upon request (Bollen, 1989; Jöreskog & Sörbom, 1993). If one can configure a suitable reduced model that excludes the target latent variable(s) of interest, then ΔR^2 can be easily calculated using Eq. (1).

It turns out that this second step – configuring a suitable reduced model that omits the effect of a target latent variable – is less straightforward than it might initially appear. The difficulties inherent in this task were well summarized by Bengt Muthén in response to a 2017 Mplus discussion forum post from a user whose attempts to calculate ΔR^2 from a SEM program output actually resulted in *negative* ΔR^2 values:

> The comparison of R-squares across models like this is a bit problematic in my view, I'm afraid, when x_1 and x_2 are latent. There is not only a question of how R-square changes when including/excluding one of them as alternatives. Because they are latent, it is also a matter of model fit - for one alternative the model may not fit at all because different restrictions are imposed on the covariances between the DV and the indicators of the x_1, x_2 factors. I know of no references [on this topic]. (B. Muthén, 2017, formatting added)

As Muthén's statement implies, the SEM framework presents several inherent challenges to the accurate calculation of ΔR^2. Ironically, these challenges result directly from the features of SEM that make the method so attractive to users: the ability to fit overidentified models featuring omitted paths and other constraints, the ability to purge error from one's manifest measures through the estimation of latent variables, and the ability to address missing data using built-in *full information maximum likelihood* (FIML; Arbuckle, 1996) estimation. The statistical machinery that makes these features possible relies on solving a system of simultaneous equations implied by every variable included in one's model. For this reason, omitting a set of target variables in order to calculate $R^2_{Reduced}$ risks fundamentally altering the model algebra in a manner that could inject bias into the results. This risk is only heightened in the presence of missing data, when the variables to be omitted in the calculation of $R^2_{Reduced}$ may also be important causes of missing data that must be conditioned on in the model in order to meet the MAR assumption and accurately estimate the model's parameters (Rubin, 1976).

As Muthén's statement also implies, references on these topics are surprisingly scarce in the SEM literature. Only a handful of articles reference ΔR^2 in SEMs, and even fewer provide guidance for its computation. For example, Graham (2008) discusses how to implement a variety of classic general linear model analyses in the SEM framework but does not explicitly present a method for computing ΔR^2. Two authors who do provide methods for computing ΔR^2 are Preacher (2006) and de Jong (1999).[1] Both of these authors suggest elegant reparameterizations of a user's model designed to, in essence, "trick" a given SEM software package into returning R-squared change. Preacher (2006) construes ΔR^2 as the squared semipartial correlation between a given predictor and an outcome variable and obtains this standardized quantity via the strategic use of single-indicator latent variables. Alternatively, de Jong (1999) coaxes increments in R-squared out of SEM software via a latent Cholesky decomposition of the variance-covariance matrix of a set of predictor variables.

Despite their effectiveness and sheer cleverness, implementing Preacher's (2006) and de Jong's (1999) methods may prove challenging for applied researchers hoping to compute ΔR^2 in their own analyses. While the mathematical rationales for these approaches may make intuitive sense to quantitative methodologists, the inherent complexity of these methods may prove intimidating to users less comfortable with SEM software defaults, single-indicator latent variable approaches, path tracing rules, and (in de Jong's case) Cholesky decompositions of variance-covariance matrices. This is particularly troubling given that the amount of syntax

[1] I wish to offer a sincere thanks to the second reviewer on this manuscript for pointing me toward this reference, which had eluded my initial literature search prior to the first submission of this paper.

required to implement either method is formidable, with ample opportunity for errors in the hands of less confident users. User specification errors may result in convergence warnings that could prove difficult to troubleshoot without a solid understanding of the mathematical logic of the models. Worse still, such errors could simply result in a misspecified model that produces no warnings at all, but that ultimately returns an incorrect result that may be easily mistaken for accurate.

Preacher's (2006) approach is arguably more straightforward to implement than the latent Cholesky decomposition proposed by de Jong (1999), and his method works perfectly for computing ΔR^2 due to the addition of a single manifest or latent variable. However, there is no obvious extension of this method for computing the change in *R*-squared due to a set of ≥ 2 predictors. Furthermore, although treating ΔR^2 as a squared semipartial correlation is mathematically and statistically equivalent to treating it as the difference in multiple R^2 due to the addition of a single predictor in an OLS regression model, experience suggests that applied researchers do not typically think about ΔR^2 in terms of semipartial correlations. With this in mind, it seems intrinsically desirable to develop more intuitive, less syntactically demanding general approaches to computing ΔR^2 in SEMs that map more directly onto the way real researchers are used to computing and conceptualizing this statistic: as the difference between a full and reduced model R^2 using the hierarchical regression approach depicted in Eq. (1).

Toward this end, the present article attempts to provide a reference for researchers on the topic of computing ΔR^2 in SEMs featuring latent variables and missing data. I begin by reviewing four possible approaches to computing ΔR^2 in SEMs that map directly onto the familiar approach of Eq. (1), clearly delineating which approaches produce accurate results under general conditions and which approaches may produce biased results in latent variable models with missing data. Then, I establish proof of concept by comparing the methods' performance in a demonstrative simulation. Following this, I describe a series of extensions of the methods and conclude with recommendations for substantive researchers and a description of three accompanying R functions (R Core Team, 2013) that can be used in concert with the lavaan package (Rosseel, 2012) to implement the direct matrix calculations described under Approach 3.

Throughout the article, I strive to present all technical material in a manner detailed enough to retain the interest of quantitative and methodological experts but accessible enough to remain useful to substantive and applied researchers. As such, the majority of matrix algebra is relegated to the appendices and to the internal arguments of two easy-to-use R functions. Knowledge of path tracing rules (see McArdle, 2005; Wright, 1934) is useful for understanding the origins of certain key equations throughout the paper, but these may just as easily be taken on faith and even skipped over without major loss of continuity.

Four approaches to computing *R*-squared differences in SEMs with latent variables and missing data

As a template for discussing ΔR^2 in SEMs, imagine that the model of Fig. 1a is the full model encompassing a researcher's hypotheses for a certain study. Assuming that the model fits the data reasonably well, the researcher may be interested in reporting R^2 measures of effect size. If the interest is in the proportion of variance explained by the set of both latent predictors, ξ_1 and ξ_2, this quantity can be easily calculated using the formula:

$$R^2_{\eta_i} = 1 - \frac{\mathrm{var}(\zeta_i)}{\mathrm{var}(\eta_i)} \tag{2}$$

where η_i is the *i*th endogenous variable (in this case, η_1), $\mathrm{var}(\eta_i)$ is its total variance, and $\mathrm{var}(\zeta_i)$ is the disturbance (residual) variance of η_i after its prediction in a structural regression (see Bollen, 1989, pp. 117–119; Jöreskog & Sörbom, 1993 pp. 26–27). Originally implemented in LISREL, this type of R^2 calculation is now performed upon request by most standard SEM software packages such as lavaan (Rosseel, 2012) and Mplus (L. K. Muthén & Muthén, 2017).

But what if the researcher is particularly interested in the increment in R^2 contributed by predictor ξ_2, above and beyond that due to ξ_1? For example, perhaps ξ_2 is the researcher's key predictor of interest (e.g., smoking habits) while ξ_1 is a control variable (e.g., other negative diet and lifestyle factors) included in the latent variable regression. Once the full model of Fig. 1a is estimated, it is easy to compute R^2_{Full} using Eq. (2). But the question remains how to appropriately compute $R^2_{Reduced}$ in order to calculate ΔR^2 via Eq. (1). In the sections that follow, I describe four potential strategies for computing the increment in R^2 due to ξ_2 in the model of Fig. 1a and compare their strengths and weaknesses.

Approach 1: Dropping ξ_2 from the model to calculate $R^2_{Reduced}$

Mirroring OLS regression, the most intuitive approach to calculating $R^2_{Reduced}$ in the model of Fig. 1 would be to simply omit the measurement model for the target variable, ξ_2, from the larger SEM and calculate $R^2_{Reduced}$ on this reduced model using Eq. (2). That is, one would omit the entire measurement model – indicators x_5 through x_8 – from the reduced model, as depicted in Fig. 1b, thereby removing the entire ξ_2 factor. The primary advantages of this approach are its simplicity, its familiarity to researchers used to implementing an analogous procedure to calculate ΔR^2 with observed variables in OLS regression, and its accuracy when the data are complete and the model of Fig. 1a is the true data generating model.

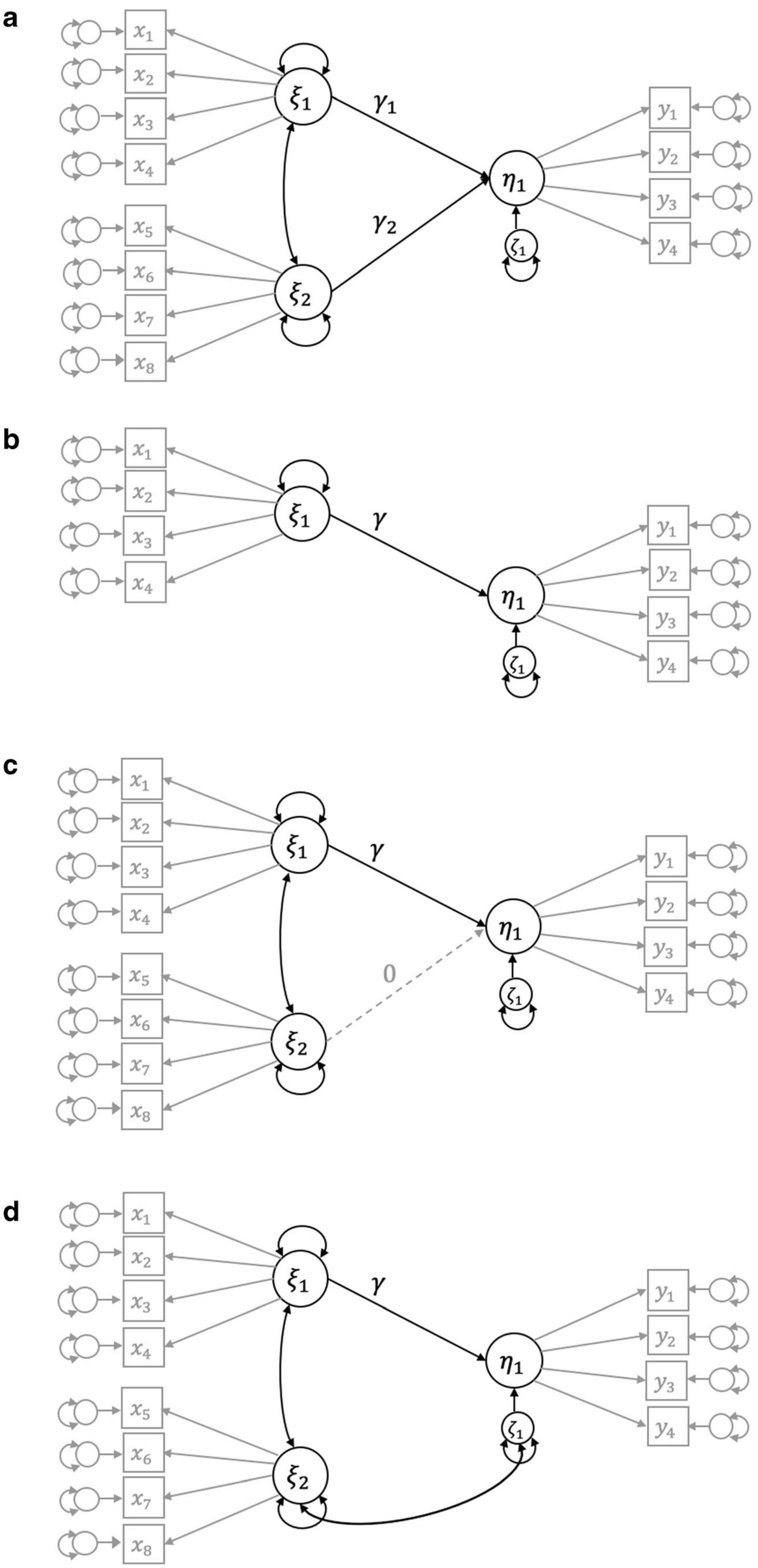


Fig. 1 (**a**) Full latent variable model and (**b**)–(**d**) possible approaches to specifying a reduced model omitting the structural regression of η_1 on ξ_2

Despite its intuitive appeal, however, this approach carries limitations. For example, even if global model fit appears acceptable, any parameter bias due to minor model misspecifications (e.g., omitting small nonzero cross-loadings or correlated uniquenesses; see B. Muthén & Asparouhov, 2012) might manifest itself differently in the full model of Fig. 1a and the reduced model of Fig. 1b, leading to estimates of R^2_{Full} and $R^2_{Reduced}$ that are, in essence, "moving targets" (B. Muthén, 2017). Furthermore, if the measurement models for ξ_1 and ξ_2 were connected by cross-loadings, removing the measurement model for factor ξ_2 in Fig. 1b would no longer be possible without biasing the parameters of the model for ξ_1, perhaps severely (Asparouhov et al., 2015; Morin et al., 2013; B. Muthén & Asparouhov, 2012), rendering questionable any estimate of $R^2_{Reduced}$ calculated from the model.

More important still, even if simple structure and conditional independence are reasonable approximations to the population model, as drawn in Fig. 1, Approach 1 may falter in the presence of missing data. As a simple example, suppose that missing data on indicators y_1–y_4 are directly caused by the measurement model for ξ_2. This could occur, for example, if individuals higher in a latent measure of smoking behavior (ξ_2) were less willing to report information related to their distal health outcomes (indicators y_1–y_4 of latent factor η_1). Furthermore, suppose that the researcher is using built-in FIML estimation to address missing data (Arbuckle, 1996).

Like multiple imputation (MI; Enders, 2010; Little & Rubin, 1987; Rubin, 1987), FIML estimation is only accurate under the assumption that the data are missing at random (MAR: Rubin, 1976). This assumption implies that the probability of missing data on reported health outcomes is uncorrelated with participants' actual health outcomes (the data they would have provided, but chose not to) after conditioning on the true causes of missing data (their level of latent smoking behavior), assumed present in the model. Stated plainly, if smokers are simply less likely than nonsmokers to report their health information in general, regardless of whether their health is good or poor, then the observed (complete data) scores obtained from smokers who opt to respond to health questions may be treated as a random subsample of possible scores drawn from the conditional distribution of health outcomes among smokers. If so, the conditional mean and variance of health outcomes will be estimated accurately in the full structural regression model of Fig. 1a. If the same can be said for all other strata of smoking behavior (e.g., if the observed health outcome data among moderate smokers represents a random sample of their possible scores, and if the same can be said for low smokers and nonsmokers, etc.), then all conditional means (y-hat values) and (residual) variances will be unbiased in the regression of Fig. 1a, and the model's coefficients will be estimated accurately.

As is well known in the missing data literature, however, if one fails to condition on the true causes of missing data in one's model, then the MAR assumption is no longer met, and any parameter bias caused by missing data will remain, regardless of whether one uses a modern missing data handling method like FIML or MI in fitting the model (see Collins et al., 2001; Rubin, 1976). That is, if latent smoking behavior (ξ_2) really is the true cause of missing data on health outcomes (η_1) and if, further, this variable is left out of the analysis model, as would be the case in the reduced model of Fig. 1b, then the conditional distributions of reported (observed, non-missing) health outcomes within each strata of remaining predictor ξ_1 (say, latent dietary habits) will not be the same conditional distributions as those estimated in the full model of Fig. 1a and will no longer be guaranteed to represent random subsamples of participants' possible health outcome scores. For this reason, any bias caused by missing data will remain in the model estimates, even if FIML estimation is switched on in one's SEM software package of choice.[2]

In sum, Approach 1 has the potential to be effective under the ideal conditions of perfectly correct model specification and complete data but may break down under misspecification or missing data. Under missing data, Approach 1 will fail if ξ_2 and its indicators represent important causes of missingness that must be included in the model in order to meet the MAR assumption. The essence of MAR is that the conditional distributions of a given outcome variable may be treated as completely random with respect to missing data only when the analyst has conditioned on the correct set of predictors (the true missing data causes). It is this imperative to include rather than omit key missing data predictors from the model that renders Approach 1 (dropping ξ_2) potentially precarious when missing data are present. Next, I describe an alternative strategy that retains the ξ_2 measurement model instead of dropping it, but restricts its structural path to η_1 to zero.

[2] An equivalent way to conceive of the MAR assumption is in terms of controlling for a covariate in multiple regression. If one envisions smoking (ξ_2) as the cause of a 0/1 missing data indicator corresponding to nonresponse on health outcomes (η_1), such that the only correlation between the missing data indicator and participants' scores on health outcomes (both observed and missing) in the population is through their common cause (smoking, ξ_2), then controlling for smoking behavior in a regression model would render the missing data indicator conditionally independent from the values of the health outcome scores that would have been obtained if all participants had provided complete data. As such, the missing data indicator does not need to be explicitly included in the conditional regression predicting health outcomes, as it exerts no influence on health outcomes after partialling out its true cause (it is *ignorable*, see Little & Rubin, 1987). If smoking is not included as a covariate in the model, however, then its influence will not be partialed out of the 0/1 missing data indicator, and this indicator will remain correlated with the missing scores on health outcomes. This is analogous to a quasi-experimental model in which a variable appears to exert a significant treatment effect until a confounding variable is entered into the analysis, rendering the influence of the spurious treatment indicator nonsignificant.

Approach 2: Fixing the $\xi_2 \rightarrow \eta_1$ structural path to zero

Aside from dropping ξ_2 from the model entirely, arguably the second-most intuitive approach to estimating $R^2_{Reduced}$ would be to retain the ξ_2 measurement model (including its exogenous covariance with ξ_1) in the reduced model but constrain the structural regression of η_1 on ξ_2 to zero. This would serve to include all variables in the reduced model instead of dropping the ξ_2 measurement model and risking potentially harmful ramifications for parameter bias and missing data estimation. Approach 2 is visually depicted in Fig. 1c, in which the structural path from $\xi_2 \rightarrow \eta_1$ is drawn as a gray, dotted line with a zero value next to it, indicating that this path may either be omitted from the model completely or, equivalently, explicitly fixed to zero in one's model syntax.

Although on the surface Approach 2 may appear to circumvent the shortcomings inherent in Approach 1, its potential benefits are undercut by a fatal flaw: if one constrains the $\xi_2 \rightarrow \eta_1$ path to zero in one's model when this path actually departs significantly from zero in the population, then this overidentifying constraint will alter the model algebra in a manner that will likely harm both local and global fit and ultimately inject parameter bias throughout that model regardless of whether or not there are missing data on any model variables (Bollen, 1989; Kenny et al., 1998; Kenny & Milan, 2012). When the measurement model for ξ_2 is also an important cause of missing data on the other model variables (e.g., indicators y_1–y_4 of η_1), omitting the $\xi_2 \rightarrow \eta_1$ path will additionally affect the ability of FIML estimation to adjust for missing data. By constraining the direct relationship between ξ_2 and η_1 to zero, the model of Fig. 1c postulates that ξ_2 is only related to η_1 indirectly through its correlation with ξ_1. As such, the FIML algorithm used to estimate the model will not be able to fully adjust the parameter estimates and implied covariances for the likely influence of missing data because the covariance information supplied to the algorithm will be fundamentally inaccurate and incomplete (this is because this model is, essentially, a saturated correlates model with an omitted correlation; see J. W. Graham, 2003).

It seems clear, then, that neither Approach 1 nor Approach 2 represents a perfect solution to the problem of computing ΔR^2 in SEMs. In the following sections, I describe two alternative approaches that allow analysts to compute ΔR^2 without encountering the problems raised by Approaches 1 and 2: (1) a direct matrix calculation of ΔR^2 using parameter estimates from the full model (Approach 3), and (2) a *saturated correlates* approach to specifying the reduced model (see J. W. Graham, 2003).

Approach 3: Direct matrix calculation of $R^2_{Reduced}$

A third approach to calculating ΔR^2 in SEMs involves direct matrix calculation of $R^2_{Reduced}$ from the model-implied covariances obtained from the full model of Fig. 1a. To lay the foundation for this approach, recall that in OLS regression, R^2 for a given model, say one's full model, may be calculated using the formulas:

$$\begin{aligned} R^2_{Full} &= \boldsymbol{\beta}'\mathbf{R}_{xx}\boldsymbol{\beta} \\ &= \boldsymbol{\beta}'\mathbf{r}_{xy} \end{aligned} \tag{3}$$

where $\boldsymbol{\beta}$ is the column vector of standardized regression weight with $\boldsymbol{\beta}'$ as its transpose, $\mathbf{R}_{xx}$ is the correlation matrix of the predictor variables, and $\mathbf{r}_{xy}$ is a column vector of correlations between each predictor and the outcome, y (Cliff, 1987; Pedhazur, 1997). Calculation of R^2_{Full} using Eq. (3) requires first calculating the standardized regression coefficients contained in $\boldsymbol{\beta}$. This may be accomplished using the formula:

$$\boldsymbol{\beta}=\mathbf{R}_{xx}^{-1}\mathbf{r}_{xy}. \tag{4}$$

These formulas imply that one can calculate R^2 and $\boldsymbol{\beta}$ from a set of sufficient statistics – in this case, the sample correlations among the x's and y – rather than the raw data. Suppose that Eqs. (3) and (4) are used to calculate R^2_{Full} and $\boldsymbol{\beta}$ based on one's full model, including all predictors and covariates of interest. Calculation of $R^2_{Reduced}$ and, in turn, ΔR^2 may be easily achieved by partitioning the correlation matrix $\mathbf{R}_{xx}$ and correlation vector $\mathbf{r}_{xy}$ into parts involving the reduced model variables and parts involving the target variables from the full model that are to be omitted in the full model.

Specifically, let $\mathbf{R}_{xx_{[r]}}$ be the correlation matrix of the predictors included in the reduced model, with the target predictors from the full model omitted, and let $\mathbf{r}_{xy_{[r]}}$ be a column vector of correlations between each predictor from the reduced model and the outcome variable, y. With these definitions in hand, it is easy to see how one may calculate $R^2_{Reduced}$ as:

$$\begin{aligned} R^2_{Reduced} &= \boldsymbol{\beta}'_{[r]}\mathbf{R}_{xx_{[r]}}\boldsymbol{\beta}_{[r]} \\ &= \boldsymbol{\beta}'_{[r]}\mathbf{r}_{xy_{[r]}} \end{aligned} \tag{5}$$

where

$$\boldsymbol{\beta}_{[r]}=\mathbf{R}_{xx_{[r]}}^{-1}\mathbf{r}_{xy_{[r]}}. \tag{6}$$

Once R^2_{Full} and $R^2_{Reduced}$ are calculated using Eqs. (5) and (6), ΔR^2 can be calculated directly via Eq. (1).

Though these concepts may seem obvious to readers well versed in matrix formulations of multiple regression, they pave the way for a direct matrix method for calculating ΔR^2 in SEM. The primary difference between direct matrix calculation of ΔR^2 in OLS regression and direct matrix calculation of ΔR^2 in SEM is that in OLS, the correlations contained in $\mathbf{R}_{xx}$ and $\mathbf{r}_{xy}$ and their partitions are observed sample correlations, whereas in SEM, these must be replaced with *model-implied correlations* among predictor and outcome variables that may be manifest, latent, or a combination of the two. This requires computing the full model-implied correlation matrix of all manifest and latent variables, as detailed in Appendix 1.

Once one has calculated the full model-implied correlation matrix, $\widehat{\mathbf{R}}_{Full}$, as described in Appendix 1, it is easy to perform direct matrix calculation of $R^2_{Reduced}$ from Eqs. 5 and 6 by simply defining $\mathbf{R}_{xx_{[r]}}$ from Eqs. 5 and 6 in terms of the model-implied correlations among the reduced-model predictors in a given structural regression of interest within one's larger SEM and defining $\mathbf{r}_{xy_{[r]}}$ as a vector of model-implied correlations between the reduced model predictors and the outcome variable.[3] In the running example, this can be accomplished by setting $\mathbf{R}_{xx_{[r]}} = \widehat{r}_{\xi_1\xi_1} = 1$ and $\beta_{[r]} = \mathbf{r}_{xy_{[r]}} = \widehat{r}_{\xi_1\eta_1}$, yielding an estimate of $R^2_{Reduced} = \beta'_{[r]}\widehat{r}^{-1}_{\xi_1\xi_1}\beta_{[r]} = \beta'_{[r]}(1)\beta_{[r]} = \beta^2_{[r]} = \widehat{r}^2_{\xi_1\eta_1}$, exactly as one would expect in a standardized bivariate regression. Assuming that R^2_{Full} has already been calculated from the full model of Fig. 1a using Eq. (2), ΔR^2 can be calculated as the simple difference defined in Eq. (1).

The matrix method of calculating $R^2_{Reduced}$ and ΔR^2 carries a key advantage over Approaches 1 and 2: this method effectively fixes in place the covariance structure implied by one's full model (e.g., the model of Fig. 1a). As a result, global fit in the reduced model will not be altered by dropping model variables, as in Approach 1, or by changing the model's algebraic structure, as in Approach 2. Equally important, any adjustments for missing data made in estimating the full model will remain intact when using the implied correlations to calculate $R^2_{Reduced}$ from the reduced model.

For these reasons, the direct matrix approach to calculating ΔR^2 solves the problems encountered in Approaches 1 and 2. The main drawback to this approach is that matrix manipulations are not necessarily intuitive to most applied researchers (particularly with larger matrices and more complex structural models than that of Fig. 1a), and these calculations are not currently implemented in standard SEM software packages. To overcome this limitation, I have written and provided an R function, rsquareCalc(), to be used in conjunction with the lavaan package (Rosseel, 2012). The function takes as input one's estimated full model fit using one of lavaan's built-in functions – e.g., lavaan() or sem() – and character strings listing the outcome variable of interest in the larger SEM and which target variables from the full model are to be omitted in calculating the reduced model. This function is provided both in Appendix 2 and as an Online Supplemental file. Additional function arguments are explained under "Extensions" below.

With the aid of rsquareCalc(), lavaan users may well find the direct matrix method for calculating ΔR^2 the simplest approach to implement. Not all researchers are proficient in R or lavaan, however. For those who prefer other software packages (e.g., Mplus), I describe an alternative *saturated correlates* approach to specifying the reduced model that will produce equivalent results to the direct matrix calculation approach without requiring users to engage in potentially error-prone matrix calculations.

Approach 4: Specifying ξ_2 as a saturated correlate in the reduced model

Approach 3 succeeds where Approaches 1 and 2 fail by holding the implied covariance structure from the full model of Fig. 1a constant when calculating $R^2_{Reduced}$ rather than altering it by dropping model variables or imposing parameter constraints. By performing direct matrix calculations on the model-implied correlations, the global fit of one's full model remains intact, and the influence of all full model variables on estimation under missing data is implicitly accounted for in the calculation of $R^2_{Reduced}$. Direct matrix calculation of $R^2_{Reduced}$ is not the only way to achieve these desirable outcomes, however. Intuitively, any reparameterization of the reduced model that retains all full model variables while holding constant the implied covariance structure and resulting global fit should produce identical results.

One simple way to accomplish this in the running example is to specify target predictor ξ_2 as a *saturated correlate* (J. W. Graham, 2003), as depicted in Fig. 1d. The saturated correlates model was originally proposed by Graham (2003) as a way of incorporating *auxiliary variables* (Collins et al., 2001) – missing data helper variables necessary for accurate and/or precise estimation under the MAR assumption but ancillary to one's substantive analysis – into FIML-estimated SEMs without altering the interpretation of the model's parameters. A saturated correlate, like ξ_2 in Fig. 1d, is specified to covary with (a) all exogenous variables in a given structural model and (b) all endogenous disturbances.

[3] Note that the x variables in $\mathbf{R}_{xx_{[r]}}$ and $\mathbf{r}_{xy_{[r]}}$ are x variables only in the sense of being regression predictors; they do not have to be restricted to the exogenous $\mathbf{x}$ model of the LISREL framework defined in Eq. (10) in Appendix 1. The predictors chosen for $\mathbf{R}_{xx_{[r]}}$ may be exogenous in the context of the larger model, endogenous in the context of the larger model, or a combination of both. This last scenario might arise, for example, if ξ_1, ξ_2, and η_1 were all predictors of a more distal outcome, say η_2, in an extended version of Fig. 1a. In this extended model, the unique contribution of ξ_2 in explaining variance in η_2 could be calculated by defining $\mathbf{R}_{xx_{[r]}}$ to include the model-implied correlations between exogenous ξ_1 and endogenous η_1, since both of these variables would serve as predictors in the regression of interest.

Correlating ξ_2 with both ξ_1 and ζ_1 serves to saturate the covariance matrix between ξ_2 and other structural model variables, leaving the global fit in the rest of the model unchanged. Thus, the saturated correlates approach to specifying the reduced model succeeds in preserving the implied covariance structure from the full model, along with its implications for global fit and missing data estimation, while reducing the structural regression portion of the model to include one-headed arrows only between the outcome variable and the reduced model (non-target) predictors. Because the amount of residual variance, var(ζ_1), leftover after η_1's prediction by ξ_1 in Fig. 1d will necessarily differ from the amount of residual variance leftover when η_1 is predicted by both ξ_1 and ξ_2 in Fig. 1a, one can calculate $R^2_{Reduced}$ from the saturated correlates model using Eq. (2) then proceed to calculate ΔR^2 as usual via Eq. (1).

The primary advantages of this approach are (1) that it can be implemented by users of any SEM software package (not just lavaan) and (2) that it does not require direct matrix manipulation. The primary disadvantage of this approach is that specifying all covariances between a given saturated correlate and all other structural model predictors can be tedious in models with many predictors, although syntax shortcuts in some programs can simplify this task (e.g., x10 WITH x1-x9 in a hypothetical model run in Mplus).[4]

Demonstrative simulation

The preceding discussion implies a set of predictions concerning when Approaches 1–4 will produce accurate results and when some of these methods may be biased. Specifically, Approach 1, dropping target variable ξ_2, should produce accurate results when the model of Fig. 1a is the true population model and the data are complete but biased results when ξ_2 is an important cause of missing data on the other model variables. Approach 2, constraining the $\xi_2 \rightarrow \eta_1$ path to zero, should produce biased results in any model in which the $\xi_2 \rightarrow \eta_1$ path is actually nonzero in the population, regardless of whether the data are complete or missing. Finally, Approaches 3 (direct matrix calculation) and 4 (specifying ξ_2 as a saturated correlate in the reduced model) should produce equivalent, accurate results regardless of whether the data are complete or missing. To test these predictions and establish proof of concept, I conducted a focused Monte Carlo simulation in R (R Core Team, 2013) using the lavaan package (Rosseel, 2012) to fit all structural models.

Data generation and simulation factors

I generated data for the simulation based on the full model of Fig. 1a. I began by simulating the latent factors, ξ_1, ξ_2, and η_1 as standard normal variates. Because the main simulation outcome was the accuracy of each of the four approaches in estimating $R^2_{Reduced}$ and ΔR^2, the first simulation factor varied was the effect size of ΔR^2. In a first set of simulation conditions, I set $\Delta R^2 = .05$, a relatively small effect size by Cohen's (1988) standards.[5] In a second set of conditions, I set $\Delta R^2 = .13$, a moderate effect size by Cohen's standards. Finally, in a third set of conditions, I set $\Delta R^2 = .26$, a large effect size by Cohen's standards.[6] The structural parameters set to obtain these values are listed in Table 1.

In addition to varying the effect size of the increment in R^2 due to ξ_2 in the latent variable model, I also varied the overall sample size, simulating one set of conditions with a small sample size of $N = 100$ and a second set of conditions with a large sample size of $N = 500$. After generating the standardized latent variables at a particular sample size, I simulated the set of manifest indicators, x_1–x_8 and y_1–y_4. Across all simulation cells, I set all factor loadings equal to $\lambda = 0.61$ and all unique factor variances equal to $(1 - \lambda^2) = 0.63$, resulting in standardized manifest indicators and reliability of $\alpha = \omega = .7$ for each four-indicator factor model (Cronbach, 1951; McDonald, 1999; McDonald & Ho, 2002).

For each unique simulation cell, I saved both a complete dataset and a copy of the same dataset with approximately 50% missingness injected on indicators y_1–y_4 of η_1 using a

[4] As one reviewer wisely pointed out, in models consisting entirely of manifest variables with missing data, the reduced model may be specified with all saturated correlates listed under the AUXILIARY subcommand in Mplus (Asparouhov & Muthén, 2008). Invoking the AUXILIARY subcommand automatically fits a saturated correlates model without requiring the user to write the syntax for the auxiliary portion of the model. This approach will not work with latent variables, however, and including all manifest indicators as saturated correlates will alter the model fit. In the model of Fig. 1, this approach would be tantamount to specifying both the common factor, ξ_2, and all unique factors loading onto its indicators as saturated correlates. By allowing all parts (common and unique) of these indicators to correlate freely with all other common and unique variables in the model, any minor misfit incurred by specifying the unique factors as conditionally independent would be lost, and the resulting model parameter estimates would be liable to shift in unpredictable ways.

[5] Cohen's (1988) arbitrary cutoff for a small R^2 effect size is, of course, .02, but I opted to simulate the slightly larger value of .05 in order to obviate potential floor effects and show more clearly the possible bias resulting from attenuation of ΔR^2 in some key simulation cells.

[6] It is worth noting that because in each set of simulated conditions the indicators of ξ_2 would be used to inject missing data on the indicators of η_1, as described below, and because the relative biasedness resulting from a given pattern of missing data under a MAR mechanism is a direct function of the degree of correlation between the true cause of missing data, e.g., ξ_2, and the variable on which data are missing, e.g., η_1 (with completely orthogonal variables resulting in MCAR rather than MAR missing data; see Collins et al., 2001; Enders, 2010; Little & Rubin, 1987), I adjusted the exogenous correlation, $r_{\xi_1\xi_2}$, in each condition instead of holding this parameter constant in order to ensure that the correlation between ξ_2 and η_1 would remain high enough to cause bias under missing data ($r_{\xi_2\eta_1} \approx .5$ in all conditions).

Table 1 Population values for the structural model used in the simulation by effect size condition

	ΔR^2Effect size		
	Small	Medium	Large
$r_{\xi_1\xi_2}$	.7	.3	.05
$\gamma{=}\gamma_1{=}\gamma_2$	.31	.38	.51
R^2_{Full}	.33	.37	.55
$R^2_{Reduced} = r^2_{\xi_1\eta_1}$	.28	.24	.29
ΔR^2	.05	.13	.26

Note: all latent variables were standardized such that $var(\xi_1) = var(\xi_2) = var(\eta_1) = 1$. For convenience, both structural regression parameters were set equal to each other

missing data generating strategy based on the MAR sinister mechanism described by Collins, Schafer, and Kam (2001).[7] I chose this type of mechanism because it is specifically designed to inject bias in the correlation between two variables, in this case ξ_1 and η_1, when the cause of missing data (the ξ_2 measurement model) is omitted from the model. If $r_{\xi_2\eta_1}$ is biased by missing data, then $R^2_{Reduced} = r^2_{\xi_1\eta_1}$ will be correspondingly biased, affecting the estimate of ΔR^2 calculated using Eq. (1).

To adapt Collins et al.'s (2001) MAR sinister mechanism to the model of Fig. 1a, I first formed sum score composites for each simulated set of exogenous indicators. Let $x_{C_1} = x_1 + x_2 + x_3 + x_4$ be the sum score composite for the indicators of ξ_1 and $x_{C_2} = x_5 + x_6 + x_7 + x_8$ be the sum score composite for the indicators of ξ_2. Then, following Collins et al. (2001), I split the dataset into $N/10$ random groups (10 groups of 10 individuals in the N = 100 conditions and 50 groups of 10 individuals in the N = 500 conditions, respectively). In each random subgroup, I computed the correlation between the exogenous composites, $r_{x_{C_1}x_{C_2}}$ and sorted the groups from the highest to lowest values of $r_{x_{C_1}x_{C_2}}$. In the half of the random subgroups that exhibited the highest correlations, $r_{x_{C_1}x_{C_2}}$, I randomly deleted 80% of the values of y_1–y_4. In the half of the random subgroups that exhibited the lowest correlations, $r_{x_{C_1}x_{C_2}}$, I randomly deleted 20% of the values of y_1–y_4. Under this mechanism, individuals with the highest observed correlations between x_{C_1} and x_{C_2} were more likely to be missing on y_1–y_4 than individuals with the lowest correlations between x_{C_1} and x_{C_2}. As a result, the correlation between ξ_1 and η_1, $r_{\xi_1\eta_1}$ should be attenuated when ξ_2 is dropped from the model under Approach 1, resulting in an inflated estimate of ΔR^2 when computed using the difference defined in Eq. (1).

In sum, the simulation consisted of a fully crossed 2 (sample size: N = 100 vs. N = 500) ×3 (effect size: ΔR^2= .05, .13, or .26) design with six unique simulation cells. Replicating each cell 1000 times yielded 6000 unique datasets, each saved twice – once with complete data and once with missing data on y_1–y_4. With four approaches to calculating the target R^2 metrics, this yielded 6000 × 2 × 4 = 48,000 different analyses.

Simulation analyses and outcomes

I computed R^2_{Full}, $R^2_{Reduced}$, and ΔR^2 according to Approaches 1–4 using the lavaan package (Rosseel, 2012) to estimate all models. I specified the reduced models for Approaches 1, 2, and 4 according to the path diagrams depicted in Fig. 1b–d, respectively. I implemented Approach 3 (direct matrix calculation) using the rsquareCalc() function described above, provided in Appendix 2. For all models fit to datasets with missing data, I implemented FIML estimation by invoking the argument missing = "fiml" in the sem() function in lavaan. The primary simulation outcomes were the average estimates of $R^2_{Reduced}$ and ΔR^2 calculated from each approach in each unique simulation cell.

Simulation results and discussion

Table 2 displays the results of the simulation. The top half of Table 2 displays the results for the N = 100 (small sample size) conditions, while the bottom half of Table 2 displays the results for the N = 500 (large sample size) conditions. Examining the table, several trends are clear. First, unsurprisingly, all methods performed better in larger samples (N = 500) than in smaller samples (N = 100). To assess the performance of each approach, begin by examining the rows of Table 2 corresponding to Approach 1, in which the measurement model for ξ_2 is dropped in the reduced model. Across

[7] The decision to inject an arbitrarily high missing data rate (50%) in the simulation was based on the simple goal of providing a clear, stark demonstration of the effects of missing data on the model parameter estimates. Because the influence of missing data percentage is straightforward and well known, with higher percentages of missing data resulting in more severe parameter bias, and because this factor was not the focus of the present study, I opted not to simulate additional conditions. However, it bears emphasizing that, as shown by Collins et al. (2001), lower rates of missing data (e.g., 25%) can also result in harmful amounts of parameter bias.

Table 2 Average simulation results by condition

	Small effect size				Moderate effect size				Large effect size			
	ΔR^2=.05		$R^2_{\xi_1}$ = .28		ΔR^2=.13		$R^2_{\xi_1}$ = .24		ΔR^2=.26		$R^2_{\xi_1}$ = .29	
	Complete	Missing	Complete	Missing	Complete	Missing	Complete	Missing	Complete	Missing	Complete	Missing
Approach:	N = 100											
1. Drop	0.08	0.15	0.30	0.29	0.14	0.20	0.26	0.24	0.27	0.33	0.30	0.26
2. Constrain	0.03	0.08	0.35	0.36	0.10	0.15	0.30	0.29	0.24	0.31	0.33	0.28
3. DM	0.08	0.12	0.29	0.32	0.14	0.16	0.26	0.28	0.27	0.28	0.30	0.31
4. Sat Cor	0.08	0.12	0.29	0.32	0.14	0.16	0.26	0.28	0.27	0.28	0.30	0.31
	N = 500											
1. Drop	0.05	0.09	0.29	0.26	0.13	0.18	0.24	0.20	0.26	0.32	0.29	0.23
2. Constrain	0.00	0.02	0.34	0.33	0.09	0.14	0.29	0.24	0.23	0.31	0.32	0.25
3. DM	0.05	0.06	0.29	0.29	0.13	0.14	0.24	0.25	0.26	0.26	0.29	0.29
4. Sat Cor	0.05	0.06	0.29	0.29	0.13	0.14	0.24	0.25	0.26	0.26	0.29	0.29

Note: ΔR^2 indicates the difference in R^2 between a model that includes both predictors, and a model that only includes ξ_1, $R^2_{\xi_1}$ indicates the R^2 in a reduced structural model that only includes ξ_1, (that is, $R^2_{Reduced}$), Drop indicates a model in which the entire factor model for ξ_2 is dropped from the model as in Fig. 1b, Constrain indicates a model in which the structural path from ξ_2 to η is constrained to zero as in Fig. 1c, DM indicates direct matrix calculation of ΔR^2 using the full model-implied correlation matrix generated by the full model of Fig. 1a, and Sat Cor indicates a model in which the ξ_2 is specified as a saturated correlate as in Fig. 1d

effect sizes and missing data mechanisms, there is a clear pattern: Approach 1 generally produces accurate estimates of both ΔR^2 and $R^2_{Reduced}$ ($R^2_{\xi_1}$ in the tables) under complete data, but biased results under missing data. As expected, dropping the ξ_2 measurement model from the reduced model resulted in attenuated estimates of $R^2_{\xi_1}$ and magnified estimates of ΔR^2.

Examining the rows of Table 2 corresponding to Approach 2, in which the $\xi_2 \rightarrow \eta_1$ path is constrained to zero, there is evident bias in the majority of simulation cells. First, restrict your attention to the analyses of complete data. Because ξ_2 is forced to correlate with η_1 only indirectly through ξ_1, the structural coefficient, γ, becomes inflated in the reduced model, resulting in inflated estimates of $R^2_{\xi_1}$ and shrunken estimates of ΔR^2 under complete data.

Turning to the Approach 2 analyses under missing data, recall that because the $\xi_2 \rightarrow \eta_1$ path is left out of this model, the full correlation between ξ_1 and η_1 is not accounted for in this model, and FIML estimation is unable to adjust for bias due to missing data as effectively as it would if ξ_2 was specified as a saturated correlate. The correlation-attenuating nature of the missing data mechanism used in the simulation somewhat counteracts the biasing effects of constraining the $\xi_2 \rightarrow \eta_1$ path to zero: whereas this overidentifying constraint tends to exert pressure toward inflating $r_{\xi_1\eta_1}$, the correlation-attenuating missing data mechanism tends to exert pressure toward deflating the very same correlation. As a result, on the surface, the estimates of ΔR^2 in some of these cells do not appear severely biased despite the very real biasing pressures exerted by both the model misspecification and missing data mechanism.

Finally, both the direct matrix approach and the saturated correlate approach produce identical, generally accurate results across all cells of the simulation, with only minor shifts in the estimates of ΔR^2 and $R^2_{\xi_1}$ under 50% missing data, regardless of mechanism. This general pattern of results was approximately the same in the N = 500 as in the N = 100 conditions, with the caveat that all estimates at this lower sample size were somewhat less accurate, on average, and this was particularly so under 50% missing data (i.e., with only N = 50 complete cases on y_1–y_4).

Extensions

Having demonstrated the accuracy (and equivalence) of the direct matrix and saturated correlates approaches to calculating ΔR^2, it is possible to envision a series of straightforward extensions to these methods. In the sections that follow, I describe how these approaches can be extended to multiple-group models, to models with dichotomous or ordinal y variables, to models with dichotomous predictor variables, and to changes in adjusted R-squared statistics as well the issue of obtaining confidence intervals for the ΔR^2 effect size. Throughout, I attempt to keep the content readable and digestible. However, some of the material covered in these sections is, admittedly and somewhat unavoidably, more technical than the material covered in the first half of this paper. For this reason, readers who are uninterested in a particular extension (e.g., those who have continuous outcomes and do not currently need to address ΔR^2 with dichotomous y variables, or those who are comfortable with SEM as a large sample framework and who, therefore, do not particularly care about extending these methods to adjusted R^2 values, etc.) may skip over any of the sections that follow with only minimal loss of continuity.

Multiple-group models

Both the direct matrix and saturated correlates approaches can be easily extended to multiple-group models. The only change required is to apply the method – either the direct matrix calculations or the saturated correlates specification of the reduced model – within each independent group. To aid in these computations, the function rsquareCalcMG(), described in Appendix 3 and provided in the Online Supplemental .R file, extends the basic calculations from the rsquareCalc() function to multiple group SEMs.

R-squared change in SEMs with dichotomous and ordinal outcomes

The focus of this article has primarily been on extending ΔR^2 to structural regression models featuring continuous latent or manifest outcome variables. Yet, if one is willing to adopt a latent probit or logit model, assuming an underlying continuous latent variable with thresholds defining the discrete observed values (as described in Long, 1997; and implemented in Mplus, see L. K. Muthén & Muthén, 2017; and lavaan, see Rosseel, 2012), then the approaches described in this article could, in theory, be applied with minimal adjustment. For example, imagine a legal psychology study in which a key outcome is jurors' guilty ($y = 1$) vs. not guilty ($y = 0$) verdicts. If one assumes a continuous, latent y^* variable underneath this dichotomous outcome, representing jurors' internal judgments of a defendant's guilt, and a threshold representing a latent tipping point (e.g., beyond a reasonable doubt) after which jurors' observed decision would switch from not guilty (0) to guilty (1), then the methods described so far may adapted to obtain *R*-squared change on the metric of this underlying latent propensity to render a guilty verdict (y^*). Although a thorough tutorial in latent variable formulations of dichotomous and ordinal outcomes is far beyond the scope of this paper, I sketch the relevant details below as they apply to computing ΔR^2 in the hope that they will be informative and helpful to readers already familiar with these models. Readers whose primary interests lie in methods for continuous outcomes may skip this section without loss of continuity.

The main difference between continuous and categorical (dichotomous, ordinal) outcome models comes in computing R^2_{Full} and $R^2_{Reduced}$. With dichotomous or ordinal outcomes, R^2 may be computed on the metric of the underlying latent variable as the ratio of the predicted variance to the total variance, with the total variance calculated as the sum of the predicted variance and a fixed residual variance set either to 1 in the probit framework or to $\pi^2/3$ in the logit framework (for details, see Bauer, 2009; Long, 1997; McKelvey & Zavoina, 1975). Luckily, Mplus performs these transformations and automatically outputs latent R^2 values for all dichotomous and ordinal outcomes when a user requests stdyx standardization. Similarly, the lavaan package will also perform these computations with a dichotomous or ordered outcome when a user sets rsquare = TRUE in either the summary() or parameterEstimates() functions, although missing data handling for categorical outcomes is currently unavailable in lavaan.

When the data are complete, computing ΔR^2 using the direct matrix approach with a dichotomous or ordinal outcome may easily be accomplished using lavaan in conjunction with the rsquareCalc() function described in Appendix 2, but with one small change: the user must set the argument conditional.x = FALSE in the function used to fit the ordinal outcome model (e.g., the lavaan or sem function). By default, this argument is set to TRUE, resulting in a conditional specification of the model that treats exogenous predictor variables as fixed and omits them from the model-implied covariance matrix.[8] Setting conditional.x = FALSE ensures that all predictor variables will be included in the final model matrices such that these can be extracted from the fit object and direct matrix computation can proceed as usual.

Computing ΔR^2 using the saturated correlates approach involves the added difficulty of correlating predictor variables with the fixed residual of an underlying latent y^* variable. When the data are complete and the user adopts the latent probit framework using WLSMV estimation in Mplus, the reduced model can be specified using the saturated correlates approach with no need for further changes. When there are missing data, however, and the user wishes to switch to a full information estimator in Mplus, specification of the reduced model becomes trickier. Users hoping to address missing data in the logistic regression framework typically invoke ESTIMATOR = ML in Mplus. However, this estimator does not allow for another variable to be correlated with the residual of the latent y^* variable.

A viable solution to this problem involves two changes to specifying the reduced model: (1) switch to the latent probit framework implemented using ESTIMATOR = BAYES, which allows more flexible specification of residual covariances, and (2) switch from a saturated correlates approach to specifying the reduced model (which produces an inadmissible covariance structure using default Bayesian estimation

[8] For further details on the `conditional.x` argument, see the `lavaan` package documentation as well as Yves Rosseel's responses on this Google group thread: https://groups.google.com/g/lavaan/c/EWqFQO3FZds?pli=1.

settings in Mplus) to Graham's (2003) other method for incorporating auxiliary variables into an analysis: the extra DV model. Note that the switch to Bayesian estimation is simply a means to the end of getting Mplus to compute the reduced model and, therefore, ΔR^2. Because the resulting estimate of ΔR^2 will be the same as it would have been using frequentist estimation, if only such an option were possible, researchers may report this effect size in conjunction with their full model results, regardless of whether they chose to fit the full model in the Bayesian or the frequentist framework.

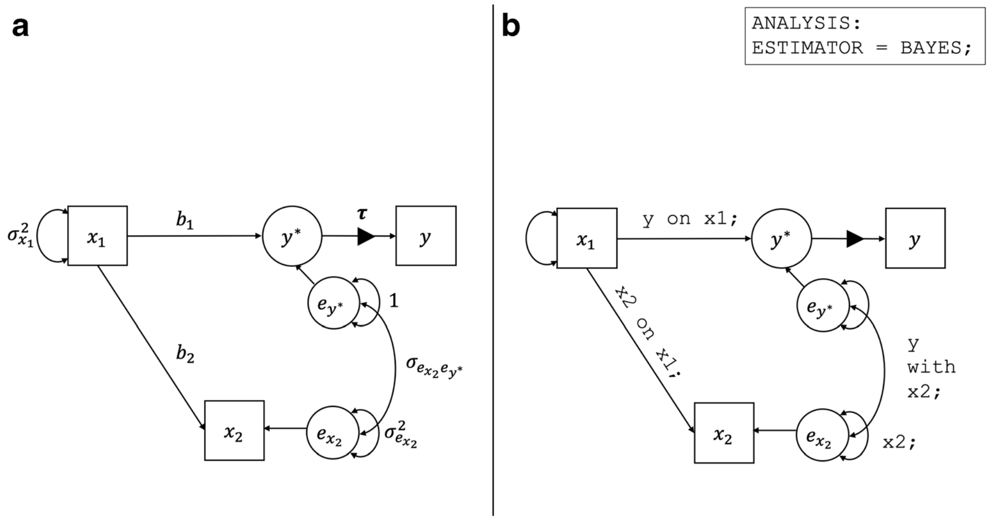
To illustrate this approach to the reduced model, briefly consider a scenario in which a researcher is predicting a dichotomous or ordinal outcome variable, y, from two predictor variables, x_1 and x_2, in a full model and wishes to obtain the ΔR^2 that occurs when x_2 is added into the model. The full model is easily specified in Mplus, but the reduced model is substantially trickier. Figure 2a and b depicts the extra DV approach to this task. In the extra DV model (Graham, 2003), all auxiliary variables (in this case, x_2) are regressed on the exogenous predictors (in this case, x_1), and their residuals are allowed to covary with the residuals of all model outcome variables. This model captures the covariances between the exogenous predictors and auxiliary variables using regressions (ON statements in Mplus) rather than exogenous covariances (WITH statements in Mplus) but is otherwise identical to the saturated correlates model and produces identical results. Figure 2a displays the statistical model, whereas Fig. 2b displays the same model with the relevant Mplus model syntax superimposed in place of the model parameters. For more details and example code, see the Online Supplemental Materials.

R-squared change in SEMs with dichotomous and ordinal predictors

The preceding section described how to extend both the direct matrix and saturated correlates approaches to models with dichotomous or ordinal outcomes. But what about models with dichotomous predictors? For example, it is extremely common for researchers to assess the ΔR^2 caused by a dichotomous experimental treatment variable (0 = control, 1 = treatment) above and beyond some set of control variables. Assessing ΔR^2 due to a dichotomous predictor is no trouble in context of standard OLS regression. But what about models with latent variables and/or missing data, as described here?

Unfortunately, dealing with dichotomous predictors is not always so straightforward in SEMs. The reasons for this are somewhat technical, but I will describe them here in brief. As described above, with dichotomous outcome variables, SEM analyses assume that the dichotomous variable is simply a coarse manifestation of a normally distributed latent variable, y^*. But standard regression models treat dichotomous predictor variables as is. This makes intuitive sense: researchers including a dummy-coded variable indicating experimental treatment, for example, are not interested in modeling a latent y^* propensity to receive treatment; they want to model the variable as an explicit, manifest predictor that yields an estimate of an explicit, manifest treatment effect.

Because it is impossible for manifest dichotomous predictors to be normally distributed, SEM software packages like Mplus and lavaan treat these predictors as fixed quantities that are excluded from the multivariate normal likelihood function invoked when using ML or FIML estimation to fit a model. When a predictor is declared as having a mean and variance and, more importantly, as explicitly covarying with other



```
ANALYSIS:
ESTIMATOR = BAYES;
```

```
y on x1;
x2 on x1;
y
with
x2;
x2;
```

Fig. 2 Extra DV approach to specifying the reduced model with a dichotomous or ordinal outcome: (**a**) Conceptual/statistical model. (**b**) Model with superimposed Mplus commands. Note that this model must be specified in the latent probit framework with ESTIMATOR = BAYES, as depicted in the box in the upper right-hand corner of panel b.

predictors and outcomes in the model, the behind-the-scenes computations employed in Mplus and lavaan surreptitiously respecify these predictors as single indicator latent variables. This brings the variables into the multivariate normal likelihood function and treats them as random rather than fixed quantities.

For continuous predictors, this poses no problem. But for dichotomous predictors, this creates serious issues. Not only is this a violation of the normality assumption, it generally leads to convergence problems that frequently cause a user's model to crash. As such, the saturated correlates model above breaks down in models with dichotomous predictors, as will the single indicator latent variable methods previously proposed in the literature by Preacher (2006) and de Jong (1999).

Fortunately, the direct matrix method will work regardless of the scaling of a given target predictor (or set of predictors). The direct matrix approach can incorporate dichotomous predictors just as well as OLS regression methods can because this approach simply borrows matrix formulas from OLS regression and extends them to more complex SEMs that may contain latent variables, missing data, or both. If all predictors contain complete or nearly complete data, such that one is willing to treat the predictors as fixed, the direct matrix approach can be used as is with the R functions provided in conjunction with the lavaan package. With substantial amounts of missing data on the exogenous predictors, however, it may be advisable to switch from FIML to multiple imputation (Rubin, 1987) for missing data handling. If so, the filled-in predictors may still be treated as fixed in the analyses conducted on each imputed dataset. In such a case, the point estimate of ΔR^2 would simply be the average of the $\Delta\widehat{R}^2$ estimates obtained by applying the provided rsquareCalc() function to each imputed dataset. If the imputed data sets are each stored as elements in a list object in R, for example, this may easily be accomplished using the lapply() function with only minimal post-processing required to average the estimates.

Change in adjusted *R*-squared

Maximum likelihood estimation and FIML for missing data are both derived from asymptotic theory under the assumption of large sample sizes. For this reason, standard SEM packages output R^2 values but not adjusted R^2 values. That is, SEM software packages compute R^2 values for each endogenous outcome using the formula from Eq. (2), with both the residual and total outcome variances implicitly computed as their respective sums of squares divided by the sample size, N. In line with standard SEM packages, the direct matrix and the saturated correlates approaches described above and tested in the demonstrative simulation follow this same convention.

However, it is well known that R^2 is a biased estimator of its true population quantity that only increases as additional predictors are added to the model, lending itself to overfitting the observed sample data (Cohen et al., 2003; Pedhazur, 1997). To counteract this tendency, OLS regression textbooks recommend the use of the *adjusted (shrunken) R-squared*, R^2_{Adj}, rather than its unadjusted counterpart. In contrast to the unadjusted R^2, R^2_{Adj} is an unbiased estimator that imposes a penalty for adding additional noisy predictors into the model that explain little or no variance in the outcome. The penalty imposed becomes more severe as the sample size becomes smaller and the number of predictors included in the model becomes larger.

Despite the norm of obtaining unadjusted R^2 values from SEM software programs, there is no reason, in principle, why a user could not tweak the procedures described above in order to compute R^2_{Adj} in place of the unadjusted R-squared values assumed so far and compute ΔR^2_{Adj} in place of ΔR^2. This is easily accomplished, as R^2_{Adj} can be written in terms of the unadjusted estimate, $\widehat{R}^2$, already provided by SEM software via the formula:

$$R^2_{Adj} = 1 - \left(\frac{N-1}{N-p-1}\right)\left(1-\widehat{R}^2\right) \tag{7}$$

where N is the sample size, and p is the number of predictors (x variables) in the structural regression model of interest. With complete data and manifest variables, this calculation will exactly reproduce the adjusted R^2 values output by standard OLS regression programs. With complete data and latent variables, this calculation will yield an estimate of the adjusted R^2 value that would have resulted from an OLS regression conducted on perfectly measured, error-free manifest measures of these same constructs. Given that perfect measurement (particularly of the predictor variables) is a tacit assumption of the OLS regression methods typically used to compute R^2_{Adj} (see Cohen et al., 2003; Pedhazur, 1997), applying this metric to structural regressions among error-free latent variables seems intuitively reasonable.

Models fitted using FIML estimation under missing data raise an interesting question about the value of N that should be used in computing R^2_{Adj} in Eq. (7). When missingness is intermittent, each variable in a larger structural model may have a different complete data N, and the pattern of missing data may exert differential effects in different parts of the larger model. Which N is the most valid one to use in computing R^2_{Adj} ? One obvious choice would be to use the overall N used in the analysis. FIML computes the overall likelihood for a given model in a case-wise manner, incorporating all cases with data on even one model variable in the likelihood computation (Enders, 2010). As a result, the overall N used in a FIML analysis represents the number of cases with partial data on any variable or combination of variables.

This choice of N is certainly intuitive and defensible and results in an estimate of R^2_{Adj} equivalent to one calculated using multiple imputation methods. In multiple imputation, pooled parameter estimates are calculated as simple averages of their values calculated in each imputed dataset (Little & Rubin, 1987; Rubin, 1987). With M imputed datasets, the pooled estimate of R^2_{Adj}, $\overline{R}^2_{Adj}$, can be shown, via simple expectation algebra, to be:

$$\begin{aligned}
\overline{R}^2_{Adj} &= E\left[1-\left(\frac{N-1}{N-p-1}\right)\left(1-\widehat{R}^2_m\right)\right] \\
&= E\left[1-\left(\left(\frac{N-1}{N-p-1}\right)-\left(\frac{N-1}{N-p-1}\right)\widehat{R}^2_m\right)\right] \\
&= E(1)-\left\{E\left(\frac{N-1}{N-p-1}\right)-E\left[\left(\frac{N-1}{N-p-1}\right)\widehat{R}^2_m\right]\right\} \\
&= 1-\left\{\left(\frac{N-1}{N-p-1}\right)-\left(\frac{N-1}{N-p-1}\right)E\left[\widehat{R}^2_m\right]\right\} \\
&= 1-\left\{\left(\frac{N-1}{N-p-1}\right)-\left(\frac{N-1}{N-p-1}\right)\overline{R}^2\right\} \\
&= 1-\left(\frac{N-1}{N-p-1}\right)\left(1-\overline{R}^2\right)
\end{aligned}$$

where $\widehat{R}^2_m$ is the estimated R^2 value from the mth imputed dataset, $\overline{R}^2$ represents the pooled estimate of R^2 averaging across all M imputed datasets, and N represents the complete data sample size after missing values have been filled in during the imputation phase. Because it is well known that the pooled estimates from multiple imputation are equivalent to the estimates produced by FIML estimation under the same model (for a discussion, see Collins et al., 2001) and because the filled-in N in each imputed dataset will be the same as the total N used in a FIML analysis, it is clear that using the total N under FIML will produce values of R^2_{Adj} equivalent to those that would be obtained under the same model using multiple imputation.

An alternative choice for N may be derived from the *fraction of missing information* (FMI, Rubin, 1987). The FMI may be interpreted as the proportion of the missing data sampling variance caused by missing data uncertainty (manifested, for example, as variance in the parameter estimates across imputed datasets when using MI). Though traditionally implemented in the multiple imputation framework, the FMI can also be calculated using FIML estimation (Savalei & Rhemtulla, 2012), and this metric is output upon request for all parameter standard errors in given model in the lavaan package. For any given parameter estimate, the FMI may be used to calculate the effective sample size, N_{Eff} (Savalei & Rhemtulla, 2012), using the formula:

$$N_{Eff} = N \times (1-FMI). \tag{8}$$

The effective N can be interpreted as the complete data N that would have resulted in a standard error of the same magnitude as the missing data standard error (Savalei & Rhemtulla, 2012). Because standard errors tend to become inflated under missing data, N_{Eff} is generally considerably smaller than the nominal N in a given analyses, making N_{Eff} a more conservative value than N to be used in calculating R^2_{Adj} using Eq. (7).

However, because N_{Eff} varies by parameter, there may be multiple effective sample sizes at work in a single FIML-estimated model, making it difficult to know which value of N_{Eff} to choose. The calculation of R^2 in Eq. (1) involves the ratio of the residual outcome variance to the total outcome variance, which encompasses both the residual variance and the predicted variance due to all regression pathways in the model. With this in mind, one particularly conservative strategy would be to calculate N_{Eff} for the residual outcome variance and for all structural regression pathways to the outcome in a given model and choose the lowest value of N_{Eff} obtained to in place of N in Eq. (7). By using the smallest effective sample size value in place of N in Eq. (7), the adjustment term incorporated in R^2_{Adj} will impose a harsher penalty on the inclusion of extra predictor variables in models with population ΔR^2 values at or close to zero.

The rsquareCalc() function described in Appendix 2 includes optional arguments that perform the calculations described in this section. If adj = TRUE, R^2_{Full} and $R^2_{Reduced}$ are replaced by their adjusted counterparts. By default, the calculations are performed on the total N used in the analysis. If effN = TRUE, the function calculates the N_{Eff} for each regression coefficient and for the residual outcome variance and uses the lowest value of N_{Eff} to compute the adjusted R^2 values.

Obtaining confidence intervals for ΔR^2

If one's goal is simply to compute a point estimate of ΔR^2 or ΔR^2_{Adj}, the methods described above are all that are needed. However, as with any effect size metric, it is good practice to compute and report a confidence interval (CI) for ΔR^2. Olkin and Finn (1995) analytically derived formulas for asymptotic CIs for R^2 values and their differences (see also, Cohen et al., 2003, p. 88). Unfortunately, multiple simulation studies have found the performance of these asymptotic CIs to be lacking in all but exceptionally large samples with normally distributed data (see e.g., Algina et al., 2007; Chan, 2009). As a result, multiple authors have suggested switching from asymptotic CIs to empirically derived CIs using bootstrap resampling methods (Efron & Tibshirani, 1993). Although multiple extensions of bootstrap CIs have been proposed, I focus here on the three types of bootstrap CIs available in Mplus and lavaan.

When requested, both Mplus and lavaan compute standard errors and confidence errors using a *nonparametric bootstrap* procedure that resamples N rows of the dataset with replacement B times creating B bootstrap datasets, where B is the number of bootstrap samples, typically 1000 or more. The user-specified model is then fitted to each of the B bootstrap datasets, and the parameter estimates are saved. In the current context, the parameter estimate of interest is ΔR^2 or ΔR^2_{Adj}. For the remainder of this discussion, I use ΔR^2 rather than ΔR^2_{Adj}, but the procedure applies equally well to both statistics. Let $\Delta\widehat{R}^2_b$ be the estimate of ΔR^2 computed on the bth bootstrap dataset and let $\Delta\widehat{R}^2$ be the estimate from the original sample data, before resampling. The *bootstrap standard error*, SE_B, is simply calculated as the standard deviation of the bootstrap estimates, $\Delta\widehat{R}^2_b$. With the bootstrap standard error in hand, analysts can easily compute the standard error-based $[(1-\alpha)\times 100]\%$ *bootstrap confidence interval* (BCI) as:

$$\text{BCI} = \left[\Delta\widehat{R}^2 - z_{\left(1-\frac{\alpha}{2}\right)} SE_B, \Delta\widehat{R}^2 + z_{\left(1-\frac{\alpha}{2}\right)} SE_B\right] \quad (9)$$

where $z_{(1-\alpha/2)}$ is the (positive) z-score critical value separating the top $\alpha/2$ proportion of a standard normoral distribution from the lower $1-\alpha/2$ proportion of the distribution (see Chan, 2009; Efron & Tibshirani, 1993; Zhang et al., 2010). For a standard 95% CI, $z_{(1-\alpha/2)} = 1.96$.

As is clear from Eq. (9), the BCI employs the empirically derived bootstrap standard error, SE_B, but still relies on standard normal distribution quantiles. The two alternative bootstrap approaches provided by Mplus and lavaan do not rely on the normal distribution theory. The first approach is the *percentile bootstrap confidence interval* (PB, Efron, 1981; Efron & Tibshirani, 1993), formed as the interval between the bootstrap estimates separating the middle $[(1-\alpha)\times 100]\%$ of the empirical bootstrap sampling distribution from the remaining $[(\alpha/2)\times 100]\%$ of the distribution in each tail. The second approach is the *bias-corrected boostrap confidence interval* (*BC*) which applies a correction for instances in which substantially more or substantially fewer than 50% of bootstrap samples lie at or below the sample estimate, $\Delta\widehat{R}^2$ (for computational details, see Efron, 1981, 1987; Efron & Tibshirani, 1993),

The performance of bootstrap CIs for ΔR^2 and related quantities has been assessed in multiple simulation studies. Algina, Keselman, and Penfield (2007) found that PB confidence intervals for ΔR^2 outperformed asymptotic CIs in terms of coverage in a variety of conditions so long as $\Delta R^2 > 0$, even with relatively small sample sizes of 200 or less. Expanding on this work, Chan (2009) found that asymptotic CIs performed well in terms of coverage and bias only with normal data but that BCIs and CIs computed using the PB and BC methods all performed well with both normal and non-normal data, except when $R_1^2 = R_2^2 = \Delta R^2 = 0$. Interestingly, Chan also found that the BCI and PB methods performed just as well as the more complicated BC approach and the accelerated BC approach proposed by Efron (1987). Zhang, Preacher, and Luo (2010) obtained similar results in a simulation comparing the BCI, PB, BC, and accelerated BC methods for computing confidence intervals for factor loadings and factor correlations in exploratory factor analysis (EFA) models. These authors found that the BCI and PB methods performed the best across conditions in terms of coverage, with the BC and accelerated BC methods sometimes performing poorly due to overcorrecting the bootstrap CI when the sample estimate of a given statistic was far from the median of the bootstrap sampling distribution.

Taken together, these studies suggest that bootstrap standard errors and CIs are a valid, if computationally intensive, method for constructing confidence intervals for ΔR^2 in regression and structural equation models. The performance of bootstrapping in conjunction with FIML-estimated SEMs is also well validated (see Enders, 2001; Wu & Jia, 2013), making these methods particularly useful in the present setting. For R users, bootstrap standard errors and CIs can be easily obtained via the direct matrix approach using the rsquareCalc.Boot() function described in Appendix 4 and demonstrated in the online supplemental material. By default, the function computes bootstrap standard errors and a confidence interval for ΔR^2 using the PB method, although the BC method may also be requested instead via an optional argument. The function can also compute these same standard errors and confidence intervals for ΔR^2_{Adj} using either the full sample N or the effective N, both with or without missing data.

For Mplus users wishing to compute a confidence interval for ΔR^2, the task is not quite as simple. Although Mplus can compute bootstrap standard errors and provide confidence intervals using either the PB or BC methods, these options can only be employed within a single model (e.g., within the full model or reduced model), whereas the method for computing ΔR^2 described above involves a difference between two models (i.e., between a full model and reduced model, specified using a saturated correlates or extra DV approach). As such, users hoping to obtain bootstrap confidence intervals for ΔR^2 from Mplus have no choice but to switch to a method for obtaining ΔR^2 from a single model. As mentioned in the introduction, two such methods are Preacher's (2006) single-indicator latent variable method, which is appropriate for computing the change in R-squared in a continuous outcome due to a single predictor variable, and de Jong's (1999) latent Cholesky decomposition method, which is appropriate for computing ΔR^2 due to any number of predictors in models with continuous, dichotomous, or ordinal outcomes.[9]

[9] Although the focus of this paper has been on implementing these methods in a frequentist framework, Mplus may also be used to obtain 95% credible intervals for ΔR^2 in conjunction with these same methods when a user sets ESTIMATOR = BAYES.

Given the necessity of switching to an alternative method in order to compute a confidence interval for ΔR^2 in Mplus, one may wonder if the saturated correlates approach to computing $R^2_{Reduced}$ adds any value above and beyond these methods. I argue that the answer is yes, for two important reasons. First, analysts whose goal is simply to compute a point estimate of the ΔR^2 effect size may do so quickly and easily using the saturated correlates approach. Second, and relatedly, even if one wishes to ultimately switch to either Preacher's (2006) or de Jong's (1999) methods in order to obtain a confidence interval for ΔR^2, computing ΔR^2 first using the simple, intuitive saturated correlates approach can provide an important check against which to compare the estimate of ΔR^2 obtained from either of these more complex approaches.

As mentioned in the introduction, both Preacher's (2006) and de Jong's (1999) methods involve a substantial amount of model syntax that must be specified correctly in order to obtain a valid estimate of ΔR^2. Because the syntax for these models is somewhat intricate, the potential for user errors in specifying these models is fairly high compared to the much simpler and less potentially error-prone approach of computing ΔR^2 as the simple difference between the full and reduced models. Computing ΔR^2 using the saturated correlates approach described above before attempting to code either Preacher's (2006) or de Jong's (1999) methods affords the critical advantage of knowing the correct answer in advance. If the point estimate of ΔR^2 obtained from Preacher's (2006) or de Jong's (1999) method matches the estimate previously computed using the saturated correlates approach, one can be confident that their model syntax is correctly specified and the resulting bootstrap standard error and confidence interval are trustworthy. But if these estimates differ, the model results should not be trusted, and the model syntax should be reexamined and debugged.

In this way, the simple method described in this paper can be helpful to researchers even when they decide to ultimately go on to implement the more complex model specification approaches proposed by Preacher (2006) and de Jong (1999). To help researchers even further in their attempts to specify these complicated models correctly, Online Supplemental Appendices A–D describe and explain Preacher's (2006) and de Jong's (1999) methods, respectively, in a step-by-step manner designed to be accessible to applied researchers with a basic knowledge of SEM. The supplemental materials also provide detailed, annotated path diagrams for each model, along with example data and syntax (.inp files) for these models in Mplus.

General discussion

Researchers wishing to make incremental validity claims are often advised to switch from observed variable regression models to latent variable models estimated in the SEM framework. Yet, until now, the literature has provided surprisingly little guidance concerning how to compute the associated ΔR^2 effect size in these models. Using the model of Fig. 1a as a template, this article described four approaches to calculating ΔR^2 in SEMs with latent variables and missing data and illustrated these approaches in a demonstrative simulation before describing a series of extensions to Approaches 3 and 4. The first two approaches to specifying the reduced model – dropping the ξ_2 measurement model or constraining the $\xi_2 \rightarrow \eta_1$ path to zero – are unsatisfactory under a wide array of circumstances, particularly (but not only) when there are missing data. The second two approaches, direct matrix calculation and the saturated correlates approach, are algebraically equivalent and produce accurate results in latent variable models both with and without missing data. This implies a clear set of recommendations: researchers should generally avoid using Approach 1, particularly under missing data. Approach 2 should never be used, since it involves a fundamentally misspecified reduced model that will virtually never provide accurate results. Instead, researchers who wish to compute ΔR^2 should do so either using direct matrix calculations or using the saturated correlates approach to specifying the reduced model.

Researchers who prefer conducting SEM analyses using the lavaan package in R can easily calculate ΔR^2 or ΔR^2_{Adj} using the rsquareCalc() function provided in Appendix 2. The rsquareCalcMG() function provided in Appendix 3 extends the basic calculations to multiple-group models, and the rsquareCalc.Boot() function provided in Appendix 4 computes bootstrap standard errors and confidence intervals for ΔR^2 or ΔR^2_{Adj} for single-group models. The online supplemental materials accompanying this article contain a set of examples demonstrating how to use each function under a variety of conditions.

For those who prefer other software packages, the saturated correlates specification of the reduced model affords an easy way to calculate $R^2_{Reduced}$ and ΔR^2 without the need to perform cumbersome matrix calculations. In the context of larger SEMs that may contain many simultaneous structural regressions, this simply involves respecifying the target (manifest and/or latent) predictors as saturated correlates in the particular structural regression of interest. All other paths in the larger SEM (including one-headed arrows from the target predictor(s) to other structural outcomes elsewhere in the model) may remain unchanged from the full model.

One benefit of Approaches 3 and 4 is that any minor misfit that may exist in one's full model is held constant when estimating $R^2_{Reduced}$ using these approaches. In essence, once one settles on a full model that exhibits acceptable (if not perfect) global fit, one is implicitly staking a claim that the model is plausibly true – that one's model, as specified, represents a plausible causal structure that could have given rise to the observed covariances (Bollen, 1989; Kenny, 1979; Pearl & Mackenzie, 2018; Wright, 1934). By calculating $R^2_{Reduced}$ from the model-implied correlations, one is effectively asking the question, "in a world in which this model was true, what would the reduced model R^2 and ΔR^2 be?" That is, "what is the increment in R-squared in the world implied by this model?" Of course, to the extent that the model is at all misspecified, one's estimate of ΔR^2 will be less accurate and, potentially, less meaningful (or even meaningless, depending on how far one's specified model diverges from reality). This could be seen as a limitation of these approaches, but this same limitation is, of course, true with respect to every single parameter in every SEM model, including the model R^2 calculations routinely reported by SEM software packages using Eq. (2).

Related to the issue of misfit, it is worth addressing an important distinction: that between R^2 effect size and SEM model fit. Some readers (particularly those who are, perhaps, newer to SEM methods) may wonder why one would not simply use SEM fit indices to compare models like the full and reduced models of Fig. 1a and b. After all, this is how SEM model comparison is typically undertaken. If latent variable ξ_2 provides incremental validity over ξ_1, could this not be demonstrated by showing that a model that includes ξ_2 provides significantly improved fit compared to a model that excludes it?

The answer to this question has to do with the nature of global SEM model fit indices. Whereas the role of model-based effect size measures like ΔR^2 is to quantify the size of some positive (nonzero) effect(s) in one's model, the sole role of SEM model fit indices is to quantify the global *mis*fit between the covariance structure observed in one's sample and the covariance structure implied by a specified model. Holding constant sample size and model complexity, such misfit is a function of two primary sources: sampling variability ("estimation error") and model misspecification ("approximation error", see Steiger, 2016). As a result, if one could fit the true, correctly specified data-generating model (such that approximation error is zero) to the population-level data (such that estimation error is zero), the model would fit perfectly regardless of the size of any particular effect specified within the model.

In this way, model fit indices from correctly specified SEMs are silent with respect to the effect sizes observed throughout a model. Taken together, then, global model fit indices and model-based effect size measures like ΔR^2 each have their own unique roles to play in the process of model evaluation. Model fit indices assess the validity of the model as a whole – its plausibility as a structure that could have generated the observed sample data – whereas ΔR^2 statistics assess of the size of key partial regression effects within the model.

In summary, references on the topic of calculating ΔR^2 in SEM have been few and far between in the existing literature, despite the obvious utility of this effect size metric in applied research (a sentiment echoed by B. Muthén, 2017). I believe this work addresses several important gaps in the existing literature. Although, as stated, Preacher (2006) and de Jong (1999) each previously proposed different methods for obtaining ΔR^2 from SEMs, to my knowledge, the current paper is the first to describe Approaches 1–4. Furthermore, neither Preacher (2006) nor de Jong (1999) discussed ΔR^2 in the context of missing data, and neither author discussed the potential pitfalls of Approaches 1 and 2 to reduced model specification. This latter point is particularly important because Approaches 1 and 2 are arguably the most intuitive ways to conceive of specifying a reduced model, making them the methods most likely to potentially be tried by real researchers at the outset. As such, one important goal of this paper is to raise awareness of the fatal shortcomings of these intuitive-but-incorrect approaches in order to discourage their future use.

The present research also contributes to the literature by proposing Approaches 3 and 4, which are arguably much more straightforward to implement than Preacher's (2006) and de Jong's (1999) methods and which map more directly onto how most substantive and applied researchers likely conceptualize ΔR^2: as the difference in R^2 values obtained from a full and reduced model. The saturated correlates model of Approach 4 is intrinsically simple to implement, whereas the matrix operations of Approach 3 are rendered simple to implement with the aid of the accompanying R functions. Finally, to my knowledge, the direct matrix method of Approach 3 is the only available way to calculate ΔR^2 or ΔR^2_{Adj} in models including dichotomous predictors.

Nonetheless, to help readers who may wish to implement Preacher's (2006) or de Jong's (1999) approaches – whether in order to obtain standard errors and CIs from Mplus or other software, or for any other reason – the online supplemental appendices accompanying this article provide a step-by-step explanation of the logical rationale behind each method, as well as comprehensive syntax files demonstrating each approach. The source articles proposing these methods are written expertly, clearly, and succinctly for a target audience of quantitative methodologists. By contrast, the online supplemental appendices provided with this paper are targeted toward making these methods digestible and understandable for an applied audience. It is my hope that the approaches described in this article, in concert with the R functions provided in the appendices and the extensive online supplemental material, will prove useful for conscientious researchers hoping to report more focused measures of R^2 effect size targeting their key predictor(s) of interest.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.3758/s13428-020-01532-y.

Open Practices Statement Code for the statistical simulation reported in this paper has been provided as a supplemental R file.

Appendix 1: Calculating the full model-implied correlation matrix of all manifest and latent variables in a general structural modeling framework

Computing ΔR^2 from the model of Fig. 1a may be a straightforward task, but not all models are so simple. SEM allows users to specify complex systems of relationships among manifest and latent variables, any of which may potentially act as predictors and outcomes in a given structural regression. If the direct matrix calculation method for computing ΔR^2 is to be useful, it must be extended to general models including both manifest and latent variables. This requires computing the full model-implied correlation matrix of all manifest and latent variables.

To achieve this, take as a starting point the well-known LISREL model equations (Jöreskog & Sörbom, 1993):

$$\mathbf{x} = \boldsymbol{\Lambda}_x\boldsymbol{\xi} + \boldsymbol{\delta}, \quad \mathbf{y} = \boldsymbol{\Lambda}_y\boldsymbol{\eta} + \boldsymbol{\epsilon}, \quad \text{and} \tag{10}$$

$$\boldsymbol{\eta} = \mathbf{B}\boldsymbol{\eta} + \boldsymbol{\Gamma}\boldsymbol{\xi} + \boldsymbol{\zeta},$$

where $\mathbf{x}$ is a $(q \times 1)$ vector of manifest indicators corresponding to the exogenous latent factors, $\mathbf{y}$ is a $(p \times 1)$ vector of manifest indicators corresponding to the endogenous latent factors, $\boldsymbol{\xi}$ is an $(n \times 1)$ vector of exogenous latent factors, $\boldsymbol{\eta}$ is an $(m \times 1)$ vector of endogenous latent factors, $\boldsymbol{\delta}$ and $\boldsymbol{\epsilon}$ respresent unique factors (measurement residuals), and $\boldsymbol{\zeta}$ represents disturbances (regression residuals) in the structural regression portion of the model. As usual, $\boldsymbol{\Lambda}_x$ and $\boldsymbol{\Lambda}_y$ are factor loading matrices, and $\mathbf{B}$ and $\boldsymbol{\Gamma}$ are matrices of structural regression coefficients.

Perhaps the simplest way to obtain the covariances among all variables – manifest and latent – in this model would be to position the various LISREL matrices within McArdle's general *reticular action model* (RAM; see McArdle, 2005; McArdle & McDonald, 1984) framework. In the RAM algebra, all manifest and latent variables are stacked in a vector, $\mathbf{v}$, all asymmetric, unidirectional relationships between manifest and latent variables in the model are contained in the matrix $\mathbf{A}$, all symmetric, bidirectional relationships (variances and covariances represented by two-headed arrows in the path diagram) are contained in the matrix $\mathbf{S}$, and all unique residuals are stacked in a vector $\mathbf{u}$. The fundamental linear equation of the RAM model is:

$$\mathbf{v} = \mathbf{A}\mathbf{v} + \mathbf{u}. \tag{11}$$

Following a result from McArdle (2005), let $j = q + p$, $k = n + m$, and $r = j + k$, and define the vectors $\mathbf{v}$ and $\mathbf{u}$ as $\mathbf{v}' = [\mathbf{x}' \;\; \mathbf{y}' \;\; \boldsymbol{\xi}' \;\; \boldsymbol{\eta}']$ and $\mathbf{u}' = [\boldsymbol{\delta}' \;\; \boldsymbol{\epsilon}' \;\; \boldsymbol{\xi}' \;\; \boldsymbol{\zeta}']$. Then, the $(r \times r)$ matrices $\mathbf{A}$ and $\mathbf{S}$ may be written in terms of the quantities defined in (1) as:

$$\mathbf{A} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \boldsymbol{\Lambda}_x & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{\Lambda}_y \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{\Gamma} & \mathbf{B} \end{bmatrix}, \quad \text{and} \tag{12}$$

$$\mathbf{S} = \begin{bmatrix} \boldsymbol{\Theta}_\delta & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Theta}_\epsilon & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{\Phi} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{\Psi} \end{bmatrix},$$

respectively, where $\boldsymbol{\Theta}_\delta$ and $\boldsymbol{\Theta}_\epsilon$ are variance-covariance matrices of the unique factors $\boldsymbol{\delta}$ and $\boldsymbol{\epsilon}$, $\boldsymbol{\Phi}$ is the variance-covariance matrix of the exogenous factors, $\boldsymbol{\xi}$, and where $\boldsymbol{\Psi}$ is the variance-covariance matrix of the disturbances, $\boldsymbol{\zeta}$. Then, following McArdle (2005), it is possible to define the inverse:

$$(\mathbf{I}-\mathbf{A})^{-1} = \begin{bmatrix} \mathbf{I} & \mathbf{0} & \boldsymbol{\Lambda}_x & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \boldsymbol{\Lambda}_y(\mathbf{I}-\mathbf{B})^{-1}\boldsymbol{\Gamma} & \boldsymbol{\Lambda}_y(\mathbf{I}-\mathbf{B})^{-1} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & (\mathbf{I}-\mathbf{B})^{-1}\boldsymbol{\Gamma} & (\mathbf{I}-\mathbf{B})^{-1} \end{bmatrix}. \tag{13}$$

The full model-implied covariance matrix of all manifest and latent variables, $\widehat{\boldsymbol{\Sigma}}_{Full}$, may then be defined as:

$$\widehat{\boldsymbol{\Sigma}}_{Full} = (\mathbf{I}-\mathbf{A})^{-1}\mathbf{S}(\mathbf{I}-\mathbf{A})^{-1'} = \begin{bmatrix} \boldsymbol{\Sigma}_{xx} & \boldsymbol{\Sigma}_{xy} & \boldsymbol{\Sigma}_{x\xi} & \boldsymbol{\Sigma}_{x\eta} \\ \boldsymbol{\Sigma}_{yx} & \boldsymbol{\Sigma}_{yy} & \boldsymbol{\Sigma}_{y\xi} & \boldsymbol{\Sigma}_{y\eta} \\ \boldsymbol{\Sigma}_{\xi x} & \boldsymbol{\Sigma}_{\xi y} & \boldsymbol{\Sigma}_{\xi\xi} & \boldsymbol{\Sigma}_{\xi\eta} \\ \boldsymbol{\Sigma}_{\eta x} & \boldsymbol{\Sigma}_{\eta y} & \boldsymbol{\Sigma}_{\eta\xi} & \boldsymbol{\Sigma}_{\eta\eta} \end{bmatrix}, \tag{14}$$

where:

$$\begin{aligned} \boldsymbol{\Sigma}_{\xi\xi} &= \boldsymbol{\Phi}, \\ \boldsymbol{\Sigma}_{\eta\eta} &= (\mathbf{I}-\mathbf{B})^{-1}\left(\boldsymbol{\Gamma}\boldsymbol{\Phi}\boldsymbol{\Gamma}' + \boldsymbol{\Psi}\right)(\mathbf{I}-\mathbf{B})^{-1'}, \\ \boldsymbol{\Sigma}_{\eta\xi} &= \boldsymbol{\Sigma}'_{\xi\eta} = (\mathbf{I}-\mathbf{B})^{-1}\boldsymbol{\Gamma}\boldsymbol{\Phi}, \\ \boldsymbol{\Sigma}_{xx} &= \boldsymbol{\Lambda}_x\boldsymbol{\Phi}\boldsymbol{\Lambda}'_x + \boldsymbol{\Theta}_\delta, \\ \boldsymbol{\Sigma}_{yx} &= \boldsymbol{\Sigma}'_{xy} = \boldsymbol{\Lambda}_y(\mathbf{I}-\mathbf{B})^{-1}\boldsymbol{\Gamma}\boldsymbol{\Phi}\boldsymbol{\Lambda}'_x \\ \boldsymbol{\Sigma}_{yy} &= \boldsymbol{\Lambda}_y\boldsymbol{\Sigma}_{\eta\eta}\boldsymbol{\Lambda}'_y + \boldsymbol{\Theta}_\epsilon, \\ \boldsymbol{\Sigma}_{\xi x} &= \boldsymbol{\Sigma}'_{x\xi} = \boldsymbol{\Phi}\boldsymbol{\Lambda}'_x, \\ \boldsymbol{\Sigma}_{\eta x} &= \boldsymbol{\Sigma}'_{x\eta} = (\mathbf{I}-\mathbf{B})^{-1}\boldsymbol{\Gamma}\boldsymbol{\Phi}, \\ \boldsymbol{\Sigma}_{\xi y} &= \boldsymbol{\Sigma}'_{y\xi} = \boldsymbol{\Phi}\boldsymbol{\Gamma}'(\mathbf{I}-\mathbf{B})^{-1'}\boldsymbol{\Lambda}'_y, \quad \text{and} \\ \boldsymbol{\Sigma}_{\eta y} &= \boldsymbol{\Sigma}'_{y\eta} = \boldsymbol{\Sigma}_{\eta\eta}\boldsymbol{\Lambda}'_y. \end{aligned} \tag{15}$$

To obtain the full model-implied correlation matrix, $\widehat{\mathbf{R}}_{Full}$, define a diagonal matrix $\mathbf{D}^{-1}$ with the main diagonal containing the reciprocals of the square roots of the diagonal entries of $\widehat{\boldsymbol{\Sigma}}_{Full}$, that is: $\text{diag}\left(\mathbf{D}^{-1}\right) = \text{diag}\left(\widehat{\boldsymbol{\Sigma}}_{Full}\right)^{-1/2}$. Then, $\widehat{\mathbf{R}}_{Full}$ can be obtained by pre- and post-multiplying $\widehat{\boldsymbol{\Sigma}}_{Full}$ by $\mathbf{D}^{-1}$ as:

$$\widehat{\mathbf{R}}_{Full} = \mathbf{D}^{-1}\widehat{\boldsymbol{\Sigma}}_{Full}\mathbf{D}^{-1} = \begin{bmatrix} \widehat{\mathbf{R}}_{xx} & \widehat{\mathbf{R}}_{xy} & \widehat{\mathbf{R}}_{x\xi} & \widehat{\mathbf{R}}_{x\eta} \\ \widehat{\mathbf{R}}_{yx} & \widehat{\mathbf{R}}_{yy} & \widehat{\mathbf{R}}_{y\xi} & \widehat{\mathbf{R}}_{y\eta} \\ \widehat{\mathbf{R}}_{\xi x} & \widehat{\mathbf{R}}_{\xi y} & \widehat{\mathbf{R}}_{\xi\xi} & \widehat{\mathbf{R}}_{\xi\eta} \\ \widehat{\mathbf{R}}_{\eta x} & \widehat{\mathbf{R}}_{\eta y} & \widehat{\mathbf{R}}_{\eta\xi} & \widehat{\mathbf{R}}_{\eta\eta} \end{bmatrix}, \tag{16}$$

where $\widehat{\mathbf{R}}_{xx}$ is defined here as the model-implied correlation matrix of the $\mathbf{x}$ measurement model indicators, in contrast to $\mathbf{R}_{xx}$ defined in Eqs. (3) and (4).

It is worth noting that partitioned matrix $\widehat{\boldsymbol{\Sigma}}_{Full}$ can either be calculated directly via the RAM equations of (12)–(14) or built up from its constituent sub-matrices calculated using the LISREL expectations defined in (15). It is also worth noting that Eqs. (12)–(16) can be rewritten to accommodate various other parameterizations of SEM. For example, to obtain the all-Y (No X) LISREL model for continuous outcomes (Jöreskog & Sörbom, 1993) implemented behind the scenes in the lavaan package (Rosseel, 2012) and in special cases of the Mplus model for continuous $\mathbf{y}^*$ variables (see B. Muthén, 2004 pp. 13–14), simply omit all matrix partitions relating to the exogenous latent variables, $\boldsymbol{\xi}$, their associated structural and measurement model parameter matrices, and their measured indicators, $\mathbf{x}$.

The calculations employed by rsquareCalc() mirror those described in this appendix. Conveniently, the full model-implied correlation matrix may be extracted from any lavaan model object via the argument what = "cor.all" in the lavInspect() function, rendering the remaining calculations quite straightforward.

Appendix 2: rsquareCalc() function

This function calculates ΔR^2 or ΔR^2_{Adj} from a single model fit in lavaan, specified in the model argument, using the direct matrix calculation method. The y argument is a character string specifying the name of the structural outcome variable of interest. The x argument is a vector of one or more character strings specifying the name(s) of the target predictor(s) of interest to be omitted from the reduced model when computing $R^2_{Reduced}$. The adj argument defaults to FALSE. If set to TRUE, the function will calculate ΔR^2_{Adj}. The effN argument defaults to FALSE. If set to TRUE, and if adj = TRUE, the function will calculate ΔR^2_{Adj} using the effective N. The silent argument defaults to TRUE. If set to FALSE, no output will be printed (this option is used in conjunction with rsquareCalc.Boot(), described in Appendix 4).

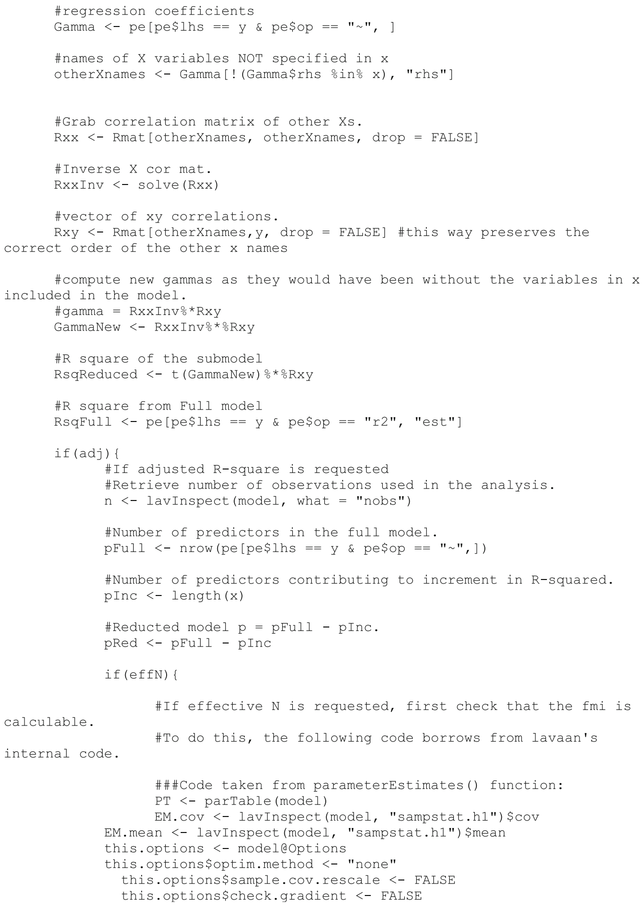
```
rsquareCalc <- function(model, y, x, adj = FALSE, effN = FALSE, silent =
FALSE){
      #model is a model fit by lavaan using e.g., the sem() or lavaan()
function.
      #y is a character vector of length 1 specifying the name of the
(single) structural outcome of interest.
      #x is a vector of one or more character strings specifying the name(s)
of the target predictor(s) of interest, to be omitted from the reduced model.
      #adj: do you want to calculate adjusted rather than unadjusted R2 and
R2 change? Defaults to FALSE.
      #effN: if TRUE, N in the adjusted R-square calculation is set to the
lowest effective N in the structural regression. Defaults to FALSE.
      #silent: if TRUE, output does not automatically print (but is returned
as invisible). Defaults to FALSE.
            #This argument is invoked when using rsquareCalc.Boot in order to
turn off default printing while taking bootstrap resamples.

      require("lavaan")

      if(!is.character(y)|length(y) != 1) stop("y must be a character vector
of length 1, specifying the name of the DV in the (manifest or latent
variable) regression of interest!")

      #parameter estimates
      pe <- parameterEstimates(model, standardized = TRUE, rsquare = TRUE)

      #correlation matrix of all variables
      Rmat <- lavInspect(model, what = "cor.all")
```



```
      #regression coefficients
      Gamma <- pe[pe$lhs == y & pe$op == "~", ]

      #names of X variables NOT specified in x
      otherXnames <- Gamma[!(Gamma$rhs %in% x), "rhs"]


      #Grab correlation matrix of other Xs.
      Rxx <- Rmat[otherXnames, otherXnames, drop = FALSE]

      #Inverse X cor mat.
      RxxInv <- solve(Rxx)

      #vector of xy correlations.
      Rxy <- Rmat[otherXnames,y, drop = FALSE] #this way preserves the
correct order of the other x names

      #compute new gammas as they would have been without the variables in x
included in the model.
      #gamma = RxxInv%*Rxy
      GammaNew <- RxxInv%*%Rxy

      #R square of the submodel
      RsqReduced <- t(GammaNew)%*%Rxy

      #R square from Full model
      RsqFull <- pe[pe$lhs == y & pe$op == "r2", "est"]

      if(adj){
            #If adjusted R-square is requested
            #Retrieve number of observations used in the analysis.
            n <- lavInspect(model, what = "nobs")

            #Number of predictors in the full model.
            pFull <- nrow(pe[pe$lhs == y & pe$op == "~",])

            #Number of predictors contributing to increment in R-squared.
            pInc <- length(x)

            #Reducted model p = pFull - pInc.
            pRed <- pFull - pInc

            if(effN){

                  #If effective N is requested, first check that the fmi is
calculable.
                  #To do this, the following code borrows from lavaan's
internal code.

                  ###Code taken from parameterEstimates() function:
                  PT <- parTable(model)
                  EM.cov <- lavInspect(model, "sampstat.h1")$cov
            EM.mean <- lavInspect(model, "sampstat.h1")$mean
            this.options <- model@Options
            this.options$optim.method <- "none"
              this.options$sample.cov.rescale <- FALSE
              this.options$check.gradient <- FALSE
              this.options$baseline <- FALSE
              this.options$h1 <- FALSE
              this.options$test <- FALSE
                  fit.complete <- lavaan(model = PT, sample.cov = EM.cov,
            sample.mean = EM.mean, sample.nobs = n, slotOptions =
this.options)
            ###

            #Check that the complete model is identified:
            if(any(eigen(lavInspect(fit.complete, what = "vcov"))$values
<0)){

                  #If the model used to estimate the fmi is non-identified,
everything is NA.
                  res <- rep(NA, 2)
                  names(res) <- c(paste("Rsquare Without ", paste0(x,
collapse = " "), collapse = ""), "RsquareChange")

            }else{

                  #Otherwise, the calculations proceed

                  #peFMI = parameter estimates with fmi
                  peFMI <- parameterEstimates(model, standardized = TRUE,
rsquare = TRUE, fmi = TRUE)

                        #Flag regression relationships with proper dv:
                        regressionflag <- peFMI$lhs == y & peFMI$op == "~"

                        #Flag the residual variance as well
                        residvarflag <- peFMI$lhs == y & peFMI$op == "~~" &
peFMI$rhs == y

                        #Subset the parameter estimates object to include all
                        #contributors to predicted and residual variance.
                        peFMI_sub <- peFMI[regressionflag|residvarflag,]

                        #Retrieve max fmi from structural model.
                        fmi <- peFMI_sub[which.max(peFMI_sub$fmi), "fmi"]

                        #Calculate the effective n.
                        effN <- n*(1-fmi)

                        #Overwrite the original n for use in subsequent
calculations.
                        n <- effN

                        #Adjusted R-square calculations.
                        multiplierFull <- (n-1)/(n - pFull - 1)
                        multiplierRed <- (n-1)/(n - pRed - 1)
                        RsqReduced <- 1 - multiplierRed*(1 - RsqReduced)
                        RsqFull <- 1 - multiplierFull*(1 - RsqFull)

                        #R square change is difference between overall R
square and reduced R square.
                        RsqChange <- RsqFull - RsqReduced
                        res <- c(RsqReduced, RsqChange)
                        names(res) <- c(paste("Rsquare Without ", paste0(x,
collapse = " "), collapse = ""), "RsquareChange")
            }

            }else{

                  #If effN == FALSE, we proceed with the overall n.

           #Adjusted R-square calculations.
                  multiplierFull <- (n-1)/(n - pFull - 1)
                  multiplierRed <- (n-1)/(n - pRed - 1)
                  RsqReduced <- 1 - multiplierRed*(1 - RsqReduced)
                  RsqFull <- 1 - multiplierFull*(1 - RsqFull)

                  #R square change is difference between overall R square and
reduced R square.
                  RsqChange <- RsqFull - RsqReduced

                  res <- c(RsqReduced, RsqChange)

                  names(res) <- c(paste("Rsquare Without ", paste0(x,
collapse = " "), collapse = ""), "RsquareChange")
            }

      }else{

            #Otherwise, simply calculate R-square change without the
adjustment terms.
            RsqChange <- RsqFull - RsqReduced
            res <- c(RsqReduced, RsqChange)
            names(res) <- c(paste("Rsquare Without ", paste0(x, collapse = "
"), collapse = ""), "RsquareChange")
      }

      #if silent printing is not requested, print the result.
      if(!silent) print(round(res, 2))

      #And return the object.
      invisible(res)
}
```

Appendix 3: rsquareCalcMG() function

This function takes the same arguments as rsquareCalc() but extends the function to calculate ΔR^2 from multiple group (MG) SEMs. If a single-group model is provided in the model argument, the function returns the same output as rsquareCalc() with adj = FALSE. However, this function relies on lavaan’s internal structure for the functions lavInspect(fit, what = “cor.all”) and parameterEstimates(fit, rsquare = TRUE), as currently implemented in lavaan version 0.6-5 at the time of this writing. If lavaan’s internal storage methods for these functions are changed in any future release, the function may no longer work as intended. For this reason, the standalone function rsquareCalc() will likely be more stable, as it relies on aspects of the parameterEstimates() function that are less idiosyncratic and less likely to be updated in future package releases. Note also that this function presumes that the target structural regression(s) from the full model are estimated in all groups in the multiple group analysis (though this parameter may be constrained to equality across groups).

```
rsquareCalcMG <- function(model, y, x){
      #model is a model fit by lavaan
      #y is a character string specifying the name of structural outcome of
interest.
      #x is a vector of one or more character strings specifying the name(s)
of the target predictor(s) of interest, to be omitted from the reduced model.

      require(lavaan)

      #parameter estimates
      pe <- parameterEstimates(model, standardized = TRUE, rsquare = TRUE)

      #rsqc = modified rsquareCalc() function, nested within this function.
      rsqc <- function(peObj, mod){

            #correlation matrix of all variables
            RmatTest <- lavInspect(mod, what = "cor.all")

            #Is RmatTest a list of correlation matrices for each of G groups?
            if(is.list(RmatTest)){

                  #If so, this will be applied to each unique block using 
lapply() below.
                  #Retrieve the correlation matrix corresponding to the 
unique block.
                  Rmat <- lavInspect(mod, what = "cor.all")[[peObj$block[1]]]

            }else{

                  #Otherwise, there is only one matrix.
                  Rmat <- lavInspect(mod, what = "cor.all")

            }

            #regression coefficients
            Beta <- peObj[peObj$lhs == y & peObj$op == "~", ]

            #names of X variables NOT specified in x (reduced model x's).
            otherXnames <- Beta[!(Beta$rhs %in% x), "rhs"]

            #Grab correlation matrix of other Xs.
            Rxx <- Rmat[otherXnames, otherXnames, drop = FALSE]

            #Inverse X cor mat.
            RxxInv <- solve(Rxx)

            #vector of xy correlations.
            Rxy <- Rmat[otherXnames,y, drop = FALSE] #this way preserves the 
correct order of the other x names

            #compute new betas as they would have been without the variables 
in x included in the model.
            #beta = RxxInv%*Rxy
            BetaNew <- RxxInv%*%Rxy

            #R square of the submodel
            Rsq <- t(BetaNew)%*%Rxy

            #R square change is difference between overall R square and 
reduced model R square.
            RsqChange <- peObj[peObj$lhs == y & peObj$op == "r2", "est"] - 
Rsq

            if(!is.null(peObj$block)){

                  #If the analysis is multiple group, add group name as part 
of the result vector.
                  res <- c(peObj$block[1], Rsq, RsqChange)
                  names(res) <- c(paste0("Group", peObj$block[1]), 
"RsquareReduced", "RsquareChange")

            }else{

                  #Otherwise, just include Reduced R-squared and R-square 
Change.
                  res <- c(Rsq, RsqChange)
                  names(res) <- c(paste("Rsquare Without ", paste0(x,
collapse = " "), collapse = " "), "RsquareChange")

            }
            return(res)
      }

      #Is analysis MG?
      if(is.null(pe$block)){

            #If not, just calculate R-squared change for the single model.
            Result <- rsqc(pe, model)

            #Results for pretty printing (rounded to 2 decimals).
            resPrint <- Result
            resPrint <- round(resPrint, 2)

            #Rename Result columns.
            names(Result) <- c("RsquareReduced", "RsquareChange")

      }else{

            #If MG, then ...
            #Split the pe object into groups (blocks).
            peList <- split(pe, pe$block)
            #Calculate Change in R-squared for each group (block).
            ResultList <- lapply(peList, rsqc, model)
            #Reformulate the list as a table.

            for(i in 1:length(ResultList)){
                  if(i == 1) Result <- ResultList[[i]] else Result <-
rbind(Result, ResultList[[i]])
            }

            #Set group colnames as "Group".
            colnames(Result)[1] <- "Group"
            #Ensure the table is saved as a data frame rather than a matrix.
            Result <- data.frame(Result)
            #Add group labels, for convenience.
            Result$GroupLabel <- model@Data@group.label
            #Change rownames to numbers, per usual.
            rownames(Result) <- as.character(1:nrow(Result))
            #Reorder columns
            Result <- Result[,c(1, 4, 2:3)]

            #Results for pretty printing (numeric columns rounded to 2
decimals).
            resPrint <- Result
            resPrint[,3:4] <- round(resPrint[,3:4], 2)
      }

      #Print pretty version.
      print(resPrint)

      #Return full version (not rounded to 2 decimals).
      invisible(Result)
}
```

Appendix 4: rsquareCalc.Boot() function

This function calculates bootstrap standard errors and CIs for ΔR^2 or ΔR^2_{Adj}, using the direct matrix calculation method. As noted in the comments in the body of the function, the code used for bootstrap CI construction is adapted from the code used in lavaan’s parameterEstimates() function. Therefore, the bootstrap CIs returned by rsquareCalc.Boot() should mirror those that would be obtained if the user requests bootstrap CIs for ΔR^2 using Preacher’s (2006) or de Jong’s (1999) methods. The function returns the estimate of ΔR^2 or ΔR^2_{Adj} along with the lower and upper bounds of the bootstrap CI (defaults to percentile bootstrap but bias-corrected CI may be requested) and the number of the analyses performed on the bootstrap resamples that successfully converged. By default, convergence means analyses that converged and did not produce inadmissible solutions (but see the postcheck argument, described below).

Because bootstrapping involves resampling from the raw data file and computing estimates from a statistical model, the first arguments in the function are modelSyn – the lavaan model syntax to be used in running the model – and Data – indicating the data frame to be used in fitting the model. The Y argument is a character string specifying the name of the structural outcome variable of interest. The X argument is a vector of one or more character strings specifying the name(s) of the target predictor(s) of interest to be omitted from the reduced model when computing $R^2_{Reduced}$. The nboot argument sets the number of bootstrap samples, B, with $B = 1000$ by default (the same default as lavaan). FUN indicates the function from the lavaan package to be used in fitting the model (defaults to sem()). The miss argument defaults to FALSE. If set to TRUE, the model is fit with arguments missing = “fiml” and fixed.x = FALSE. The adj argument defaults to FALSE. If set to TRUE, the function will calculate ΔR^2_{Adj}. The effN argument defaults to FALSE. If set to TRUE, and if adj = TRUE, the function will calculate ΔR^2_{Adj} using the effective N. The postcheck argument defaults to TRUE, indicating that estimates from analyses of bootstrap samples that produce inadmissible solutions or Heywood cases are automatically discarded (treated as NA). The seed argument allows the user to supply an option seed for the pseudo-random number generator used to take the bootstrap resamples, thereby ensuring a reproducible result. The parallel argument defaults to FALSE. If set to TRUE, parallel processing is initiated using the snowfall package (Knaus et al., 2009). Note that the rlecuyer package will also need to be installed to use this option (Sevcikova & Rossini, 2019). ncpus defaults to 2, indicating that two processors will be used for parallel processing if parallel = TRUE. conflevel indicates the confidence level desired for the CI (default is .95 for a 95% CI). The argument bc defaults to FALSE. If set to TRUE, bias-corrected bootstrapping will be performed instead of percentile bootstrapping. The argument … indicates additional arguments to be passed to FUN.

```
rsquareCalc.Boot <- function(modelSyn, Data, Y, X, nboot = 1000, FUN =
lavaan::sem, miss = FALSE, adj = FALSE, effN = FALSE, postcheck = TRUE, seed
= NULL, parallel = FALSE, ncpus = 2, conflevel = .95, bc = FALSE, ...){
      #modelSyn is lavaan model syntax.
      #Data is the dataset to be used in fitting the model and bootstrapping.
      #Y is a character vector of length 1 containing the name of the outcome
(Y) variable in the structural regression of interest.
      #X is a character vector of length >= 1 indicating the target
predictor(s) omitted from the reduced model.
      #nboot is the number of bootstrap resamples. Defaults to 1000, same as
lavaan's bootstrapping default.
      #FUN is the function from the lavaan package to be used in fitting the
model. sem() is the default.
      #miss: is there missing data and, if so is fiml estimation desired,
with fixed.x set to TRUE?
      #adj: do you want to calculate adjusted rather than unadjusted R2 and
R2 change? Defaults to FALSE.
      #effN: if TRUE, N in the adjusted R-square calculation is set to the
lowest effective N in the structural regression. Defaults to FALSE.
      #postcheck: if TRUE (default), bootstrap estimates are discarded from
analyses of bootstrapped datasets that produce inadmissible solutions/Heywood
cases.
```

```
      #seed: defaults to NULL. If a seed is specified, the pseudo random
number generating seed is set using set.seed(seed).
      #parallel: is parallel processing desired? Defaults to FALSE.
      #ncpus: if parallel = TRUE, how many cores do you wish to call for
parallel processing? Defaults to 2.
      #conflevel: confidence level for the interval. Defaults to .95 for a
95% bootstrap CI.
      #bc: bias-corrected boostrap confidence interval is requested if bc =
TRUE. Default is FALSE, yielding a percentile bootstrap CI.
      #... optional arguments to FUN.

      #require the lavaan package
      require("lavaan")
      if(!is.character(Y)|length(Y) != 1) stop("Y must be a character vector
of length 1, specifying the name of the DV in the (manifest or latent
variable) regression of interest!")

      #N is the number of rows in the dataset.
      N <- nrow(Data)

      #If a user-specified seed is provided, set the seed to ensure a
replicable analysis.
      if(!is.null(seed)) set.seed(seed)

      #Matrix of bootstrap row indices.
      #nrow = N, and every column contains the indices for one bootstrap
resample.
      #Pre-generating the indices in this way ensures that the random number
seed
      #provides replicable results, even when using parallel processing.
      indMat <- matrix(sample(1:N, nboot*N, TRUE), nrow = N, ncol = nboot)

      #Function to pass to sapply() or sfClusterApplyLB():
      r2bootcalc <- function(colnum){
            #colnum is an index giving the column of the indMat being used
            #at a particular iteration.

            #Print colnum, giving the bootstrap sample currently under
analysis.
            print(paste0("Bootstrap Sample: ", colnum))

            #Take column of indices:
            ind <- indMat[,colnum]

            #Subset data:
            bootData <- Data[ind,]

            if(miss == TRUE){

                  #If missing data estimation is requested, set missing =
"fiml" and fixed.x = FALSE:
                  bootMod <- try(FUN(model = modelSyn, data = bootData,
missing = "fiml", fixed.x = FALSE, ...))

            }else{

                  #Try to fit user model to bootstrapped data:
```

```
                  bootMod <- try(FUN(model = modelSyn, data = bootData, ...))

            }


            if(class(bootMod) == "try-error" | lavInspect(bootMod, what =
"converged") == FALSE){

                  #If the model does not converge, results is NA:
                  R2ch <- NA

            }else if(postcheck){

                  if(!lavInspect(bootMod, what = "post.check")){
                        #If the user requests postcheck option (the default),
models with Heywood cases are discarded:
                        R2ch <- NA

                  }else{

                              #Generate rsquare change:
                              R2ch <- rsquareCalc(bootMod, Y, X, adj, effN,
silent = TRUE)["RsquareChange"]
                  }

            }else{

                  #In all other conditions, generate rsquare change:
                  R2ch <- rsquareCalc(bootMod, Y, X, adj, effN, silent =
TRUE)["RsquareChange"]

            }

            #Return R2ch:
            return(R2ch)
      }


      if(parallel){
            #If parallel computing is required, load the snowfall package:
            require("snowfall")

            #Initialize parallel computing:
            sfInit(parallel = TRUE, cpus = ncpus)

            #Load lavaan on the nodes:
            sfLibrary(lavaan)

            #Export relevant objects to the nodes:
            sfExport(list = c("modelSyn", "Data", "Y", "X", "nboot", "FUN",
"miss", "adj", "effN", "postcheck", "indMat", "rsquareCalc"))

            #deltaR2Boot is the vector resulting from unlisting the cluster
lapply (load balanced) function results:
            deltaR2Boot <- unlist(sfClusterApplyLB(x = 1:nboot, fun =
r2bootcalc))
```

```
            #Stop parallel computing
            sfStop()
      }else{

            #For non-parallel computing, simply use sapply:
            deltaR2Boot <- sapply(X = 1:nboot, FUN = r2bootcalc)

      }

      #Internal function from the boot package:
      norm.inter <- function (t, alpha){
          t <- t[is.finite(t)]
          R <- length(t)
          rk <- (R + 1) * alpha
          if (!all(rk > 1 & rk < R))
              warning("extreme order statistics used as endpoints")
          k <- trunc(rk)
          inds <- seq_along(k)
          out <- inds
          kvs <- k[k > 0 & k < R]
          tstar <- sort(t, partial = sort(union(c(1, R), c(kvs, kvs +
              1))))
          ints <- (k == rk)
          if (any(ints))
              out[inds[ints]] <- tstar[k[inds[ints]]]
          out[k == 0] <- tstar[1L]
          out[k == R] <- tstar[R]
          not <- function(v) xor(rep(TRUE, length(v)), v)
          temp <- inds[not(ints) & k != 0 & k != R]
          temp1 <- qnorm(alpha[temp])
          temp2 <- qnorm(k[temp]/(R + 1))
          temp3 <- qnorm((k[temp] + 1)/(R + 1))
          tk <- tstar[k[temp]]
          tk1 <- tstar[k[temp] + 1L]
          out[temp] <- tk + (temp1 - temp2)/(temp3 - temp2) * (tk1 -
              tk)
          cbind(round(rk, 2), out)
      }

      #Fit model on original sample data with missing data specifications, as
above:
      if(miss){
            estMod <- try(FUN(model = modelSyn, data = Data, missing =
"fiml", fixed.x = FALSE, ...))
      }else{
            estMod <- try(FUN(model = modelSyn, data = Data, ...))
      }

      #r2change est from original sample data:
      Est <- rsquareCalc(estMod, Y, X, adj, effN, silent =
TRUE)["RsquareChange"]

      #Bootstrap standard error is standard deviation of bootstrap estimates:
      bootSE <- sd(deltaR2Boot, na.rm = TRUE)

      ###Next lines are adapted from parameterEstimates() function in lavaan:
      alpha <- (1 + c(-conflevel, conflevel))/2
      if(bc){
            zalpha <- qnorm(alpha)
            w <- qnorm(sum(na.omit(deltaR2Boot) <
Est)/length(na.omit(deltaR2Boot)))
            a <- 0
            adj.alpha <- pnorm(w + (w + zalpha)/(1 - a *(w + zalpha)))
            qq <- norm.inter(na.omit(deltaR2Boot), adj.alpha)

      }else{
            qq <- norm.inter(na.omit(deltaR2Boot), alpha)
      }

      boot.CI.Lower = qq[1,2]
      boot.CI.Upper = qq[2,2]
      ###

      names(boot.CI.Lower) <- names(boot.CI.Upper) <- NULL

      #Number of converged bootstrap replications = number of elements of
bootstrap result vector that are not NA:
      converged <- length(which(!is.na(deltaR2Boot)))

      #save results:
      res <- data.frame(Est = Est, bootSE = bootSE, boot.CI.Lower =
boot.CI.Lower, boot.CI.Upper = boot.CI.Upper, bootConvergence = converged)
      rownames(res) <- NULL
      #return results:
      return(res)
}
```

References

Algina, J., Keselman, H. J., & Penfield, R. D. (2007). Confidence intervals for an effect size measure in multiple linear regression. *Educational and Psychological Measurement*, *67*(2), 207–218. https://doi.org/10.1177/0013164406292030

Arbuckle, J. N. (1996). Full information estimation in the presence of incomplete data. In *Advanced structural equation modeling.* (pp. 243–277). Lawrence Erlbaum Associates. Inc.

Asparouhov, T., & Muthén, B. (2008). *Auxiliary variables predicting missing data*. http://statmodel2.com/download/AuxM2.pdf

Asparouhov, T., Muthén, B., & Morin, A. J. S. (2015). Bayesian structural equation modeling with cross-loadings and residual covariances: Comments on Stromeyer et al. *Journal of Management*, *41*(6), 1561–1577. https://doi.org/10.1177/0149206315591075

Bauer, D. J. (2009). A Note on comparing the estimates of models for cluster-correlated or longitudinal data with binary or ordinal outcomes. *Psychometrika*, *74*(1), 97–105. https://doi.org/10.1007/s11336-008-9080-1

Bollen, K. A. (1989). *Structural equations with latent variables*. Wiley.

Chan, W. (2009). Bootstrap standard error and confidence intervals for the difference between two squared multiple correlation coefficients. *Educational and Psychological Measurement*, *69*(4), 566–584. https://doi.org/10.1177/0013164408324466

Cliff, N. (1987). *Analyzing multivariate data*. Harcourt Brace Jovanovich.

Cohen, J. (1988). *Statistical power analysis for the behavioral sciences* (2nd ed.). Lawrence Erlbaum Associates, Inc.

Cohen, J., Cohen, P., Aiken, L. S., & West, S. G. (2003). *Applied multiple regression/correlation analysis for the behavioral sciences* (3rd Ed.). Lawrence Erlbaum Associates, Inc.

Cole, D. A., & Preacher, K. J. (2014). Manifest variable path analysis: Potentially serious and misleading consequences due to uncorrected measurement error. *Psychological Methods*, *19*(2), 300–315. https://doi.org/10.1037/a0033805

Collins, L. M., Schafer, J. L., & Kam, C. M. (2001). A comparison of inclusive and restrictive strategies in modern missing data procedures. *Psychological Methods*, *6*(4), 330–351. https://doi.org/10.1037/1082-989X.6.4.330

Cronbach, L. J. (1951). Coefficient alpha and the internal structure of tests. *Psychometrika*, *16*(3), 297–334. https://doi.org/10.1007/BF02310555

de Jong, P. F. (1999). Hierarchical regression analysis in structural equation modeling. *Structural Equation Modeling*, *6*(2), 198–211. https://doi.org/10.1080/10705519909540128

Efron, B. (1981). Nonparametric standard errors and confidence intervals. *Canadian Journal of Statistics*, *9*(2), 139–158. https://doi.org/10.2307/3314608

Efron, B. (1987). Better bootstrap confidence intervals. *Journal of the American Statistical Association*, *82*(397), 171–185. https://doi.org/10.1080/01621459.1987.10478410

Efron, B., & Tibshirani, R. (1993). *An Introduction to the bootstrap*. Chapman & Hall.

Enders, C. K. (2001). The impact of nonnormality on full information maximum-likelihood estimation for structural equation models with missing data. *Psychological Methods*, *6*(4), 352–370. https://doi.org/10.1037/1082-989X.6.4.352

Enders, C. K. (2010). *Applied missing data analysis*. Guilford Press.

Graham, J. M. (2008). The general linear model as structural equation modeling. *Journal of Educational and Behavioral Statistics*, *33*(4), 485–506. https://doi.org/10.3102/1076998607306151

Graham, J. W. (2003). Adding missing-data-relevant variables to FIML-based structural equation models. *Structural Equation Modeling: A Multidisciplinary Journal*, *10*(1), 80–100. https://doi.org/10.1207/S15328007SEM1001_4

Hunsley, J., & Meyer, G. J. (2003). The incremental validity of psychological testing and assessment: Conceptual, methodological, and statistical issues. *Psychological Assessment*, *15*(4), 446–455. https://doi.org/10.1037/1040-3590.15.4.446

Jöreskog, K. G., & Sörbom, D. (1993). *LISREL 8*. Scientific Software International, Inc.

Kenny, D. A. (1979). *Correlation and causality*. Wiley.

Kenny, D. A., Kashy, D. A., & Bolger, N. (1998). Data analysis in social psychology. In D. Gilbert, S. Fiske, & G. Lindzey (Eds.), *The handbook of social psychology* (pp. 233–265). McGraw-Hill.

Kenny, D. A., & Milan, S. (2012). Identification: A non-technical discussion of a technical issue. In R. H. Hoyle (Ed.), *Handbook of structural equation modeling* (pp. 145–163). The Guilford Press.

Kirk, R. E. (1996). Practical significance: a concept whose time has come. *Educational and Psychological Measurement*, *56*(5), 746–759. https://doi.org/10.1177/0013164496056005002

Knaus, J., Porzelius, C., Binder, H., & Schwarzer, G. (2009). Easier parallel computing in R with snowfall and sfCluster. *The R Journal*, *1*(1), 54–59.

Ledgerwood, A., & Shrout, P. E. (2011). The trade-off between accuracy and precision in latent variable models of mediation processes. *Journal of Personality and Social Psychology*, *101*(6), 1174–1188. https://doi.org/10.1037/a0024776

Little, R. J. A., & Rubin, D. B. (1987). *Statistical analysis with missing data*. Wiley.

Long, J. S. (1997). *Regression models for categorical and limited dependent variables*. Sage.

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 psychometrics: A festschrift for Roderick P. McDonald* (pp. 225–273). 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. https://doi.org/10.1111/j.2044-8317.1984.tb00802.x

McDonald, R. P. (1999). *Test theory: A unified treatment*. Lawrence Erlbaum Associates, Inc.

McDonald, R. P., & Ho, M.-H. R. (2002). Principles and practice in reporting structural equation analyses. *Psychological Methods*, *7*(1), 64–82. https://doi.org/10.1037/1082-989X.7.1.64

McKelvey, R. D., & Zavoina, W. (1975). A statistical model for the analysis of ordinal level dependent variables. *The Journal of Mathematical Sociology*, *4*(1), 103–120. https://doi.org/10.1080/0022250X.1975.9989847

Morin, A. J. S., Marsh, H. W., & Nagengast, B. (2013). Exploratory structural equation modeling. In R. O. Hancock, G. R., & Mueller (Ed.), *Structural equation modeling: A second course* (2nd Ed., pp. 395–436). Information Age Publishing.

Muthén, B. (2004). *Mplus Technical Appendices.* Muthén & Muthén. https://www.statmodel.com/download/techappen.pdf

Muthén, B. (2017). *Mplus Discussion >> R-Square Change*. http://www.statmodel.com/discussion/messages/11/24407.html?1503529120

Muthén, B., & Asparouhov, T. (2012). Bayesian structural equation modeling: A more flexible representation of substantive theory. *Psychological Methods*, *17*(3), 313–335. https://doi.org/10.1037/a0026802

Muthén, L. K., & Muthén, B. (2017). *Mplus user's guide. Eighth edition.* Muthén & Muthén.

Olkin, I., & Finn, J. D. (1995). Correlations redux. *Psychological Bulletin*, *118*(1), 155–164. https://doi.org/10.1037/0033-2909.118.1.155

Pearl, J., & Mackenzie, D. (2018). *The book of why: The new science of cause and effect.* Basic Books.

Pedhazur, E. J. (1997). *Multiple regression in behavioral research: Explanation and prediction.* (3rd Ed.). Wadsworth.

Preacher, K. J. (2006). Testing complex correlational hypotheses with structural equation models. *Structural Equation Modeling*, *13*(4), 520–543. https://doi.org/10.1207/s15328007sem1304_2

R Core Team. (2013). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing. http://r-project.org/

Rosseel, Y. (2012). lavaan : An R package for structural equation modeling. *Journal of Statistical Software*, *48*(2), 1–36. https://doi.org/10.18637/jss.v048.i02

Rubin, D. B. (1976). Inference and missing data. *Biometrika*, *63*(3), 581–592. https://doi.org/10.2307/2335739

Rubin, D. B. (1987). *Multiple imputation for nonresponse in surveys.* Wiley.

Savalei, V. (2018). A comparison of several approaches for controlling measurement error in small samples. *Psychological Methods*, Advanced Online Publication. https://doi.org/10.1037/met0000181

Savalei, V., & Rhemtulla, M. (2012). On obtaining estimates of the fraction of missing information from full information maximum likelihood. *Structural Equation Modeling*, *19*(3), 477–494. https://doi.org/10.1080/10705511.2012.687669

Sechrest, L. (1963). Incremental validity: A recommendation. *Educational and Psychological Measurement*, *23*(1), 153–158. https://doi.org/10.1177/001316446302300113

Sevcikova, H., & Rossini, T. (2019). *rlecuyer: R Interface to RNG with Multiple Streams* (R package version 0.3-5). https://cran.r-project.org/package=rlecuyer

Steiger, J. H. (2016). Notes on the Steiger–Lind (1980) handout. *Structural Equation Modeling: A Multidisciplinary Journal*, *23*(6), 777–781. https://doi.org/10.1080/10705511.2016.1217487

Wang, Y. A., & Eastwick, P. W. (2020). Solutions to the problems of incremental validity testing in relationship science. *Personal Relationships*, *27*(1), 156–175. https://doi.org/10.1111/pere.12309

Westfall, J., & Yarkoni, T. (2016). Statistically controlling for confounding constructs is harder than you think. *PLoS ONE*, *11*(3), 1–23. https://doi.org/10.1371/journal.pone.0152719

Wright, S. (1934). The method of path coefficients. *The Annals of Mathematical Statistics*, *5*(3), 161–215. http://www.jstor.org/stable/2957502

Wu, W., & Jia, F. (2013). A new procedure to test mediation with missing data through nonparametric bootstrapping and multiple imputation. *Multivariate Behavioral Research*, *48*(5), 663–691. https://doi.org/10.1080/00273171.2013.816235

Zhang, G., Preacher, K. J., & Luo, S. (2010). Bootstrap confidence intervals for ordinary least squares factor loadings and correlations in exploratory factor analysis. *Multivariate Behavioral Research*, *45*(1), 104–134. https://doi.org/10.1080/00273170903504836

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.