# Assessing measurement invariance with moderated nonlinear factor analysis using the R package OpenMx

Kolbe, L.; Molenaar, D.; Jak, S.; Jorgensen, T.D.

**DOI**
10.1037/met0000501

**Publication date**
2024

**Document Version**
Final published version

**Published in**
Psychological Methods

**License**
CC BY-NC-ND

AMERICAN PSYCHOLOGICAL ASSOCIATION

Psychological Methods


ISSN: 1082-989X

2024, Vol. 29, No. 2, 388–406
https://doi.org/10.1037/met0000501

# Assessing Measurement Invariance With Moderated Nonlinear Factor Analysis Using the R Package `OpenMx`

Laura Kolbe[1], Dylan Molenaar[2], Suzanne Jak[1], and Terrence D. Jorgensen[1]
[1]Department of Child Development and Education, University of Amsterdam
[2]Department of Psychology, University of Amsterdam

***Abstract***

Assessing measurement invariance is an important step in establishing a meaningful comparison of measurements of a latent construct across individuals or groups. Most recently, moderated nonlinear factor analysis (MNLFA) has been proposed as a method to assess measurement invariance. In MNLFA models, measurement invariance is examined in a single-group confirmatory factor analysis model by means of parameter moderation. The advantages of MNLFA over other methods is that it (a) accommodates the assessment of measurement invariance across multiple continuous and categorical background variables and (b) accounts for heteroskedasticity by allowing the factor and residual variances to differ as a function of the background variables. In this article, we aim to make MNLFA more accessible to researchers without access to commercial structural equation modeling software by demonstrating how this method can be applied with the open-source R package `OpenMx`.

***Translational Abstract***

In the field of psychology, questionnaires or tests are often used to measure latent constructs like cognition or attitude. In order to meaningfully compare observed scores derived from these measurement instruments, it is crucial that the construct is measured equivalently across individuals. This condition is also referred to as measurement invariance. Most recently, moderated nonlinear factor analysis (MNLFA) has been proposed as a method to assess measurement invariance. The advantage of this method over more traditional methods is that it assesses measurement invariance in a more flexible way. The majority of guidelines on how to perform MNLFA, however, involve commercial statistical software. In this article, we demonstrate how MNLFA can be applied in R. Our aim is to make MNLFA more accessible to researchers or practitioners without access to commercial statistical software.

*Keywords:* moderated nonlinear factor analysis, measurement invariance, R Tutorial, differential item functioning



The field of psychology is dominated by the use of questionnaires or tests that measure latent constructs like cognition, attitude, and personality. The observed scores derived from these measurement instruments are often used for decisions about, for example, which applicant is best suited for a position, whether a child is suffering from an anxiety disorder, or to what extent an intervention has improved a patient's well-being. Given the importance of these decisions, it is crucial that the construct is measured equivalently across individuals, groups, or over time. This condition is often referred to as *measurement invariance* (Meredith, 1993). If measurement invariance does not hold, observed differences between individuals or groups may reflect differential measurement and not true differences on the latent construct of interest. Assessing measurement invariance has thus become an important step in psychometric research and applications. Measurement invariance is commonly assessed by fitting a latent variable model to the data obtained from questionnaires or tests with multiple items. A measurement instrument is invariant if for each item the observed score at any given level of the latent construct is not affected by any background variables. For example, the expected observed score of an item on a social anxiety questionnaire should be the same across boys and girls who have the same level of social anxiety. An item that fails to meet this condition of measurement invariance indicates differential item (or indicator) functioning (DIF; Mellenbergh, 1989).

Measurement invariance is often examined with respect to a grouping variable using multiple-group confirmatory factor analysis (MGCFA; Vandenberg & Lance, 2000). In MGCFA, a confirmatory factor analysis (CFA) model is estimated for each group and invariance constraints are imposed on the parameter estimates (i.e., factor loadings, intercepts, and residual variances) in order to assess increasingly restrictive levels of measurement invariance. If an omnibus null hypothesis ($H_0$) of a specific level of measurement invariance is rejected, follow-up tests can be performed in order to explore which

This article was published Online First July 4, 2022.

Laura Kolbe https://orcid.org/0000-0002-4285-3939

This work was partly supported by the Dutch Research Council (NWO), Project 016.Veni.195.457, awarded to Terrence D. Jorgensen, and partly supported by Project VI.Vidi.201.009, awarded to Suzanne Jak.

Correspondence concerning this article should be addressed to Laura Kolbe, Department of Child Development and Education, University of Amsterdam, 1001 NG, Amsterdam, the Netherlands. Email: l.kolbe@uva.nl

items function differently across the groups. One strength of MGCFA is that all parameters, such as common-factor means and variances, can differ between groups. However, the accompanying limitation is that MGCFA is only designed to assess measurement invariance across a single categorical background variable. In light of this limitation, Bauer (2017) proposed moderated nonlinear factor analysis (MNLFA) as a more flexible alternative to MGCFA. The MNLFA approach examines measurement invariance in a single-group CFA model by means of parameter moderation. In contrast to MGCFA, the MNLFA approach accommodates the assessment of measurement invariance across multiple continuous and categorical background variables simultaneously.

Several empirical validation and simulation studies have examined the performance of MNLFA as a measurement invariance assessment tool (Bauer, 2017; Bauer et al., 2020; Kolbe et al., 2021). These studies showed that the method performs well both in small and large samples and with categorical and continuous data. Specifically, the results of the studies indicated that MNLFA yields unbiased parameter estimates, minimizes Type I error rates, and has acceptable to high power. This method effectively detects true violations of measurement invariance and avoids detecting negligible violations, particularly when using regularization (Bauer et al., 2020). Given its flexibility and good performance, MNLFA seems to be a promising method for evaluating measurement invariance for a great variety of researchers. But until now, the majority of available guidelines on how to perform this method involve commercial structural equation modeling (SEM) software (i.e., Mplus and SAS; see Bauer, 2017). There seems to be a lack of documentation on applying this method in open-source SEM software, which is more widely available for the global community of researchers. Performing MNLFA for measurement invariance assessment may therefore not be straightforward for researchers without access to Mplus or SAS.

This article presents a tutorial on assessing measurement invariance through MNLFA with the R (R Core Team, 2018) package `OpenMx` (Boker et al., 2011).[1] Our aim is to make MNLFA more accessible for any researcher by providing a detailed guideline on performing the method in this open-source SEM software. We will demonstrate MNLFA with a two-factor model and two background variables (categorical and continuous), but it can easily be applied to single-factor models, structural models, or models with fewer or more background variables. In the next section, we introduce the concept of measurement invariance. We then provide a brief explanation of the MNLFA approach, followed by a step-by-step guide for assessing measurement invariance with this method using `OpenMx`. Lastly, we offer concluding remarks regarding the use of MNLFA and address latest developments for measurement invariance assessment with open-source software packages.

## Background

### Measurement Invariance

In this section, we give a formal definition of measurement invariance that will be useful for understanding how it can be assessed with the MNLFA approach. Measurement invariance is said to hold if the distribution of observed item responses is not affected by any variables other than the latent constructs of interest (Mellenbergh, 1989). Given that the latent constructs are known, the definition can be mathematically expressed as

$$f(X \mid T, V) = f(X \mid T) \tag{1}$$

where $f(\cdot)$ denotes the probability distribution of a set of observed variables $X$ (e.g., items) measuring a set of one or more latent constructs $T$ and $V$ is a set of one or more background variables such as age or gender. This mathematical expression states that measurement invariance holds if the distribution of $X$ depends only on the latent construct(s) $T$ and is invariant with respect to background variable(s) $V$. Note that the condition of measurement invariance still allows for a relationship between $X$ and $V$, but it does preclude a direct effect of $V$ on the distribution of $X$ other than through its influence on $T$.

If measurement invariance does not hold, the observed item responses depend not only on the latent construct(s) but also directly on the background variable(s). In other words, a violation of measurement invariance indicates that the relationship between the observed item responses and the latent construct(s) differs as a function of the background variable(s). An item that violates measurement invariance is said to show DIF. A distinction can be made between full and partial invariance, where full invariance implies that all items of a test or questionnaire are measurement invariant and partial invariance implies that measurement invariance only holds for a subset of items and some items show DIF. Under partial invariance, groups or individuals can still be validly compared on the measurement of the latent construct as long as DIF is correctly detected and modeled.

Different levels of measurement invariance for CFA models have been defined (Horn & McArdle, 1992; Meredith, 1993; Steenkamp & Baumgartner, 1998). Consider a multidimensional factor model in which the item responses $X$ serve as indicators for the common factors $T$. This model can be specified as

$$\boldsymbol{x}_i = \boldsymbol{\tau} + \boldsymbol{\Lambda}\boldsymbol{t}_i + \boldsymbol{\varepsilon}_i. \tag{2}$$

Here $\boldsymbol{x}_i$ is a $p \times 1$ vector of $p$ observed indicator scores, $\boldsymbol{\tau}$ is a $p \times 1$ vector of indicator intercepts, and $\boldsymbol{\Lambda}$ is a $p \times r$ matrix containing factor loadings of $r$ common factors with means $\boldsymbol{\alpha}$ and covariance matrix $\boldsymbol{\Psi}$. Moreover, $\boldsymbol{t}_i$ is a $r \times 1$ vector of common-factor scores for individual $i$ and $\boldsymbol{\varepsilon}_i$ is a $p \times 1$ vector of residual scores with variances of $\boldsymbol{\theta}$. The different levels of measurement invariance with respect to $V$ ordered from least to most restrictive are

1. *Configural invariance*: implies equal factor structures across $V$,
2. *Metric invariance*: additionally implies equal factor loadings across $V$,
3. *Scalar invariance*: additionally implies equal indicator intercepts across $V$,

[1] The open-source R packages `mnlfa` (Robitzsch, 2019), `GPCMlasso` (Schauberger, 2021), and `regDIF` (Belzak, 2021) can be used to estimate MNLFA models as well, but their implementations are currently limited to unidimensional item response theory (IRT) models. Note that MNLFA can also be estimated with open-source Bayesian software like Stan (Stan Development Team, 2021). An R script demonstrating how to estimate MNLFA with Stan is available on our Open Science Framework project https://osf.io/6cyxt/.

4. *Strict invariance*: additionally implies equal residual variances across $V$.

The model for the observed indicator scores and the measurement-invariance conditions easily generalize to cases with a unidimensional factor model.

One of the traditional methods to evaluate measurement invariance with respect to a categorical background variable (e.g., group membership) is MGCFA (Vandenberg & Lance, 2000). In MGCFA, the data are divided into two independent groups and a CFA model as shown in Equation 2 is estimated for each group. Each group thus has its own set of model parameters, which can be denoted as $\boldsymbol{\tau}^{(1)}$, $\boldsymbol{\Lambda}^{(1)}$, and $\boldsymbol{\theta}^{(1)}$ for Group 1 and $\boldsymbol{\tau}^{(2)}$, $\boldsymbol{\Lambda}^{(2)}$, and $\boldsymbol{\theta}^{(2)}$ for Group 2. Measurement invariance can then be assessed by comparing the fit of models with and without increasingly restrictive equality constraints on the parameters across the grouping variable. Because this method relies on splitting the data into two groups, it is best suited to a single categorical background variable. Alternative methods for assessing measurement invariance have been proposed that allow for including multiple categorical and continuous background variables simultaneously. Among these alternatives are restricted factor analysis (RFA; Oort, 1992), multiple indicator multiple cause (MIMIC; Jöreskog & Goldberger, 1975), and MNLFA (Bauer & Hussong, 2009) models. The difference between these methods and MGCFA is that the data are aggregated over the groups. We therefore refer to these methods as *single-group methods* (Kolbe et al., 2021), the most flexible of which is MNLFA, described below. For details about the differences between the single-group methods see Bauer (2017) and Kolbe et al. (2021).

## Moderated Nonlinear Factor Analysis

In the MNLFA approach, a CFA model is estimated in which background variables are included as moderator variables. Figure 1 illustrates an example of a multidimensional MNLFA model containing two common factors, $T_1$ and $T_2$, measured by five indicators each. The idea of parameter moderation is demonstrated conceptually with the arrow pointing from $V$ to the measurement model. In the following paragraphs, we consider a single background variable for ease of understanding, but MNLFA also allows for a set of multiple background variables.

More formally, the MNLFA model for continuous indicators $X$ can be expressed as

$$\boldsymbol{x}_i = \boldsymbol{\tau}_i + \boldsymbol{\Lambda}_i \boldsymbol{t}_i + \boldsymbol{\varepsilon}_i, \tag{3}$$

where all parameters and notation remain defined as before in Equation 2, with the exception that all model parameters now have subscripts $i$. These subscripts are of special interest because they indicate that the values of the parameters can differ over individuals as a function of the background variable $V$. In fact, all parameters of the MNLFA model can be allowed to differ across values of any observed variable, including residual (co)variances $\boldsymbol{\Theta}_i$, common-factor means $\boldsymbol{\alpha}_i$, and common-factor (co)variances $\boldsymbol{\Psi}_i$. Note that in this tutorial, we only consider linear relationships between the parameters and background variables, but other functional forms (e.g., quadratic, interaction) can be modeled as well (Bauer, 2017; Bauer et al., 2020).

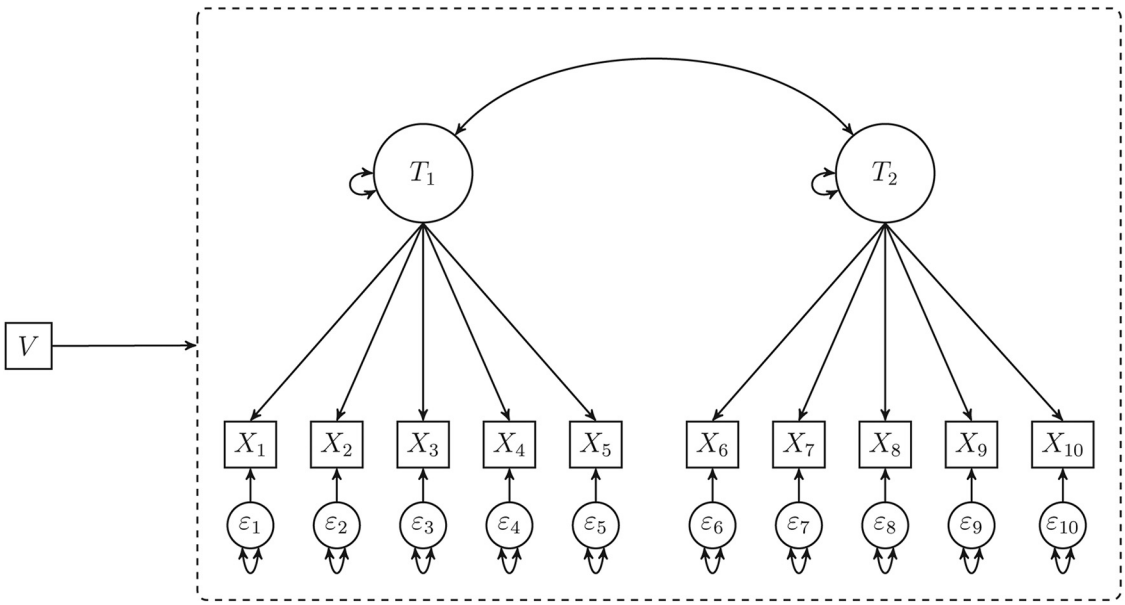
The MNLFA model presumes configural invariance, but other levels of measurement invariance can be evaluated by testing whether $V$ moderates any intercepts, factor loadings, or residual variances. More specifically, to accommodate any violations of scalar invariance, the vector of indicator intercepts $\boldsymbol{\tau}_i$ can be modeled as

$$\boldsymbol{\tau}_i = \boldsymbol{\tau}_0 + \boldsymbol{b} v_i, \tag{4}$$

where $\boldsymbol{\tau}_0$ is a $p \times 1$ vector of baseline intercepts, $\boldsymbol{b}$ is a $p \times 1$ vector of linear effects of the background variable on the intercepts, and $v_i$ is individual $i$'s score on the background variable. A nonzero element in $\boldsymbol{b}$ reflects a linear change in the intercept associated with $v_i$, indicating a violation of scalar invariance (i.e., uniform DIF).

**Figure 1**

*An Example MNLFA Model for Assessing Measurement Invariance*




*Note.* The variable $V$ may have an effect on all parameters in the model that is represented in the dashed border.

Similarly, to accommodate violations of metric invariance, each column of the matrix of factor loadings $\boldsymbol{\Lambda}_i$ can be modeled as a function of $v_i$

$$\boldsymbol{\Lambda}_i = \boldsymbol{\Lambda}_0 + \boldsymbol{C}v_i, \tag{5}$$

where $\boldsymbol{\Lambda}_0$ is a $p \times r$ matrix of baseline factor loadings and $\boldsymbol{C}$ is a $p \times r$ matrix of linear effects of $V$ on the factor loadings. A nonzero element in $\boldsymbol{C}$ reflects a linear change in the factor loadings associated with $v_i$, indicating a violation of metric invariance (i.e., nonuniform DIF).

The same idea can be adopted to accommodate violations of strict invariance. In order to prevent negative values of the residual variances, the indicators' residual variances $\boldsymbol{\theta}_i$ can be expressed as exponential functions of $v_i$

$$\boldsymbol{\theta}_i = \boldsymbol{\theta}_0 \exp(\boldsymbol{d}v_i), \tag{6}$$

where $\boldsymbol{\theta}_0$ is a $p \times 1$ vector of baseline residual variances and $\boldsymbol{d}$ is a $p \times 1$ vector containing the effects of $v_i$ on the residual variances. A nonzero element in $\boldsymbol{d}$ indicates a violation of strict invariance.

In addition to measurement parameters, common-factor means, variances, and covariances may also be moderated by the background variable $V$. The vector of means of the common factors $\boldsymbol{\alpha}_i$ may be written as a linear function of $v_i$

$$\boldsymbol{\alpha}_i = \boldsymbol{\alpha}_0 + \boldsymbol{g}v_i, \tag{7}$$

where $\boldsymbol{\alpha}_0$ is a $r \times 1$ vector containing the baseline common-factor means and $\boldsymbol{g}$ is a $r \times 1$ vector containing the linear effects of $v_i$ on the common-factor means. Moreover, the variances of the common factors may be moderated by the background variable. For example, the common-factor variance $\psi_{(T_1T_1)i}$ of common factor $T_1$ may be written as a log-linear function of $v_i$

$$\psi_{(T_1T_1)i} = \psi_{(T_1T_1)0} \exp(h_{(T_1T_1)}v_i). \tag{8}$$

Here, $\psi_{(T_1T_1)0}$ is the baseline common-factor variance and $h_{(T_1T_1)}$ reflects the direct effect of the background variable on the common-factor variance. A nonzero $h$ indicates that the common-factor variance differs across different levels of the background variable (i.e., common-factor heteroskedasticity). Similar to the model for the residual variances, a log-linear function is considered for the common-factor variances in order to prevent obtaining negative values. The baseline common-factor means and variances are commonly fixed at zero and one, respectively, in order to establish the origin and scale of the common factors.

Bauer (2017) suggested to model the covariance between common factors indirectly through a Fisher's $z$ transformation to impose bounds of −1 and 1 on the corresponding correlation. So, in order to obtain the effect of the background variable on the covariance between common factors $T_1$ and $T_2$, the Fisher-transformed correlation between the factors $\zeta_{(T_1T_2)i}$ can be written as a linear function of $v_i$

$$\zeta_{(T_1T_2)i} = \zeta_{(T_1T_2)0} + h_{(T_1T_2)}v_i, \tag{9}$$

where $\zeta_{(T_1T_2)0}$ is the baseline Fisher-transformed correlation between $T_1$ and $T_2$ and $h_{(T_1T_2)}$ reflects the direct effect of the background variable on the Fisher-transformed correlation. This Fisher-transformation can be inverted in order to obtain the correlation $\rho_{(T_1T_2)i}$ with bounds of −1 and 1

$$\rho_{(T_1T_2)i} = \frac{\exp(2\zeta_{(T_1T_2)i}) - 1}{\exp(2\zeta_{(T_1T_2)i}) + 1} \tag{10}$$

and transformed to the covariance $\psi_{(T_1T_2)i}$ between the two common factors

$$\psi_{(T_1T_2)i} = \psi^{1/2}_{(T_1T_1)i}\rho_{(T_1T_2)i}\psi^{1/2}_{(T_2T_2)i}. \tag{11}$$

The same approach could be applied to covariances between the residual factors of the indicators if present in the model.

Whereas RFA and MIMIC can only model specific parameters as functions of the background variable (i.e., common-factor means, indicators' intercepts, and factor loadings), MNLFA also allows for unique- and common-factor (co)variances to vary as functions of the background variable. The MNLFA approach can thus be conceptualized as an extended RFA or MIMIC model in which (co)variances are not necessarily homoskedastic across different levels of the background variable. This makes MNLFA as flexible as MGCFA when the background variable is dichotomous, yet more flexible because MNLFA also allows for multiple categorical and continuous background variables to be included in a single model. For more details on the MNLFA model (e.g., parameter equations for situations with multiple background variables), see Bauer (2017).

## Tutorial

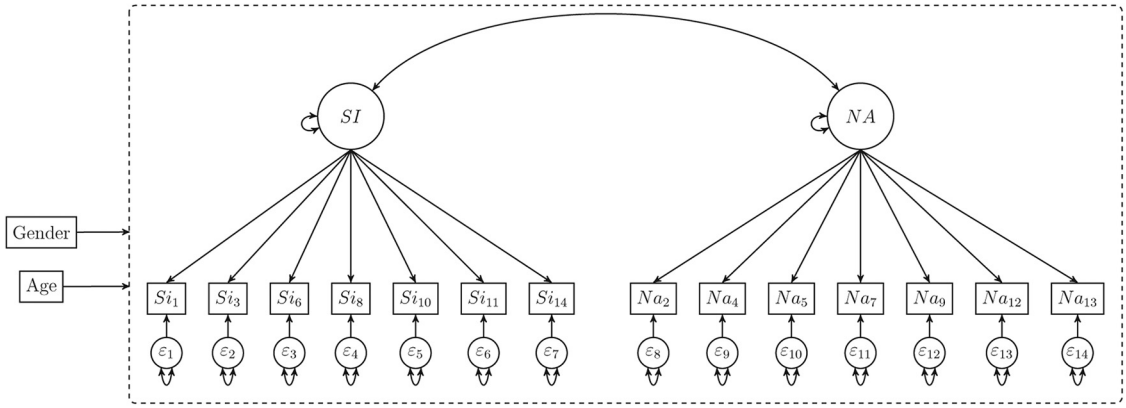
The empirical data that we used for this tutorial were gathered by Denollet et al. (2013). The data contain observed scores on the DS14 (Denollet, 2005) in a sample of 541 patients with coronary artery disease. The DS14 is a widely used instrument for the assessment of the Type D personality and consists of 14 items, of which seven measure negative affectivity and seven measure social inhibition. All items have five ordered response categories (0 = *false*, 1 = *rather false*, 2 = *neutral*, 3 = *rather true*, 4 = *true*). In addition to the observed scores on the DS14, the data also contain the dichotomous variable gender and the continuous variable age measured in years. These two variables were used as background variables in this tutorial.

The MNLFA model considered in this tutorial is shown in Figure 2. The model includes two common factors, social inhibition (SI) and negative affectivity (NA), each measured by seven indicators. The two common factors are allowed to covary and their means and variances are fixed at zero and one, respectively, in order for the model to be identified. The background variables gender and age are included in the model as moderators, and depending on the level of measurement invariance may or may not moderate the indicators' intercepts and factor loadings. In almost all MNLFA models in this tutorial, the common-factor means, common-factor variances, common-factor covariance, and indicators' residual variances are allowed to vary as a function of gender and age.[2] Such models are also referred to as *heteroskedastic MNLFA models* (see Kolbe et al., 2021).

In the next section of the article, we demonstrate how to evaluate whether the DS14 is measurement invariant with respect to gender

[2] An exception is the configural model, which requires additional identification constraints. This will be explained in Step 3a of the tutorial.

**Figure 2**
*The MNLFA Model for the DS14 Dataset*




*Note*. The background variables gender and age may have an effect on all parameters in the model represented in the dashed border.

and age using MNLFA. We provide a step-by-step tutorial for assessing full measurement invariance (i.e., assessing measurement invariance of all indicators simultaneously), selecting anchor indicators, and evaluating partial invariance (i.e., testing each indicator separately for measurement invariance). We focus on how to detect violations of scalar and metric invariance (i.e., uniform and nonuniform DIF, respectively). Such violations are commonly examined with separate omnibus tests of scalar and metric invariance, but can also be assessed simultaneously rather than separately with a single omnibus test (Muthén & Asparouhov, 2002; Putnick & Bornstein, 2016; Stark et al., 2006). In this single omnibus test, the violations of scalar and metric invariance are examined simultaneously by comparing the fit of a configural model (that allows for uniform and nonuniform DIF in all indicators) to the fit of a scalar model (that does not allow for uniform or nonuniform DIF in any indicators). Full R scripts for replicating the results of this tutorial can also be found on our Open Science Framework project https://osf.io/6cyxt/.

Before we present all of the steps of the tutorial, we would like to make two important comments. First, although the measurement level of the DS14 items is ordinal, we treat the items as continuous in the present tutorial because our aim is merely to illustrate MNLFA and not to draw substantive conclusions. Please note that if MNLFA were to be used for drawing substantive conclusions with respect to ordinal data, an ordinal MNLFA model may be more suitable (see Extensions section). For more details on treating ordinal items as continuous, see Rhemtulla et al. (2012) and Robitzsch (2020b). Rhemtulla et al. (2012) discussed conditions under which 5-point scales might be treated as continuous, and Robitzsch (2020b) elucidated why it might always be defensible to do so. Second, there is no straightforward way of assessing configural invariance (nor overall data–model fit in general) with MNLFA yet, because MNLFA does not allow for distinct factor structures with respect to the background variable(s). We will therefore assume in this tutorial that the configural-invariance condition holds. In practice, it is important to evaluate configural invariance prior to the assessment of metric, scalar, and strict invariance, because the assessment of other levels of measurement invariance may lead to false conclusions if configural invariance does not hold (Jorgensen, 2017).

## Step 1: Install and Load `OpenMx`

This tutorial illustrates how measurement invariance can be examined with MNLFA using `OpenMx` (Version 2.19.5; Neale et al., 2016). This package can be used for matrix algebra optimization, SEM, and other statistical estimation methods. The user needs to install the package on the computer once, then load it into the work space each time a new R session is started:

```
> install.packages("OpenMx")
> library(OpenMx)
```

Any dependencies are automatically installed when running the syntax above. Note that R Version 3.5 or higher is required[3] for the latest version of this package to work. The core function of the `OpenMx` package necessary to create an MNLFA model is the `mxModel()` function. This function builds an object for a statistical model containing (among other information) the data, matrices, algebraic expressions, fit functions, and expectations for the model.

## Step 2: Load and Prepare Data

The data gathered by Denollet et al. (2013) is available in the `mokken` package (Van der Ark, 2007). This R package can be installed and loaded as follows:

```
> install.packages("mokken")
> library(mokken)
```

In order to load the DS14 data and convert it to a data frame, the user can run the following lines of R code:

```
> data("DS14", package="mokken")
> DS14 <- data.frame(DS14)
```

The dataset `DS14` is now loaded into the user's work space. The `head()` function can be used to inspect the first six rows of

[3] This may have changed since the time of this writing. Check https://CRAN.R-project.org/package=OpenMx for the current requirements.

the data. The data contain gender (`Male`), age (`Age`), and item scores on the DS14 questionnaire (`Si1.`, `Na2`, `Si3.`, `Na4`, `Na5`, `Si6`, `Na7`, `Si8`, `Na9`, `Si10`, `Si11`, `Na12`, `Na13`, `Si14`).

For ease of interpretation, the two negatively worded items (`Si1.` and `Si3.`) can be recoded prior to the measurement invariance assessment:

```
> DS14$Si1 <- 4 - DS14$Si1.
> DS14$Si3 <- 4 - DS14$Si3.
```

After running the above syntax, the `DS14` data frame now also contains the recoded versions of these two items, named `Si1` and `Si3`. Similar to the other SI items, a higher score on these recoded items now indicates a higher level of social inhibition. In addition to the item scores, the data contain scores on the variables gender (`Male`) and age (`Age`). A score of `Male = 1` represents a male patient and `Male = 0` a female patient. The `Age` variable is measured in years and can be standardized in order for an easier interpretation of its effects on the model parameters:

```
> DS14$Age <- (DS14$Age - mean(DS14$Age))/
>              sd(DS14$Age)
```

For convenience, the item scores are reordered such that the first seven items reflect social inhibition and the second seven items reflect negative affectivity:

```
> DS14 <- DS14[,c("Male","Age","Si1","Si3",
>                 "Si6","Si8","Si10","Si11",
>                 "Si14", "Na2","Na4","Na5",
>                 "Na7", "Na9","Na12","Na13")]
```

After the dataset has been prepared for the measurement invariance assessment, the user should convert the data to an `MxData` object:

```
> mxdata1 <- mxData(observed=DS14,
>                   type="raw")
```

The `mxData()` function constructs an object with additional information allowing it to be processed in the `mxModel()` function. By specifying `observed=DS14` and `type="raw"`, the function reads in the raw data stored in `DS14`. Alternatively, summary statistics could be analyzed.

Finally, the user can save the number and names of the observed variables serving as indicators of the factors in the MNLFA model:

```
> manVars <- colnames(DS14[,-c(1,2)])
> nv <- length(manVars)
```

The names of the indicators stored in `manVars` are required for one of the arguments of the `mxModel()` function, which will be shown later in this section.

## Step 3: Assess Full Measurement Invariance

After preparing the data, the user can start with performing an omnibus test of full measurement invariance with respect to the background variables. That is, the user can test the $H_0$ that none of the indicators function differently with respect to `Male` and `Age`. Full measurement invariance can be assessed with MNLFA on metric, scalar, or strict levels. In this step of the tutorial, we will focus only on simultaneously assessing full scalar-and-metric invariance. In the omnibus test of scalar-and-metric invariance, the fit of an unconstrained *configural model* is compared with the fit of a constrained *scalar model*. In the configural model, the direct effects of the background variables on the indicators' intercepts and factor loadings are all freely estimated, whereas in the scalar model these effects are all fixed to zero. If the scalar model fits the data significantly worse than the configural model at a chosen $\alpha$ level, the $H_0$ of full scalar-and-metric invariance is rejected and follow-up tests can be performed to evaluate which specific indicators exhibit (non)uniform DIF with respect to the background variables (Steps 4 and 5).

### *Step 3a: Specify and Fit the Configural Model*

In order to fit this model to the empirical data with `OpenMx`, we put all model parameters into matrices using the `mxMatrix()` function. Most often, the following six arguments will be specified for each matrix:

`type`: requires a character string indicating the matrix type. In this tutorial, we use `"Diag"`, `"Full"`, and, `"Symm"` matrices.

`nrow`: refers to the number of rows of the matrix.

`ncol`: refers to the number of columns of the matrix.

`free`: indicates which elements of the matrix can be freely estimated (`TRUE` or `T`) or are fixed parameters (`FALSE` or `F`).

`values`: reflects the values of the elements in the matrix. If an element is freely estimated, it reflects the starting value. If an element is not freely estimated, it reflects the fixed value.

`name`: refers to the user-specified name of the matrix which is used within `OpenMx` when performing an operation on this matrix.

The syntax below creates three MxMatrix objects for the indicator intercepts of the configural model:

```
> matT0 <- mxMatrix(type="Full", nrow = 1,
>                   ncol=nv,
>                   free=TRUE,
>                   values = 1,
>                   name="matT0")
> matB1 <- mxMatrix(type="Full", nrow = 1,
>                   ncol=nv,
>                   free=TRUE,
>                   values = 0,
>                   name="matB1",
> matB2 <- mxMatrix(type="Full", nrow = 1,
>                   ncol=nv,
>                   free=TRUE,
>                   values = 0,
>                   name="matB2"
```

The matrix `matT0` is a full matrix containing the baseline intercepts $\tau_0$. All baseline intercepts are freely estimated with starting values of one by setting `free=TRUE` and `values=1`. Matrix `matB1` and `matB2` are full matrices containing the direct effects of the

background variables Male and Age, respectively, on the intercepts. These direct effects reflect uniform DIF, represented by *b*. In the configural model, the effects of Male and Age on the intercepts are freely estimated with starting values of zero by setting free=TRUE and values=0. By giving the matrices names using the name argument, we can refer to these matrices in upcoming syntax.

Similar lines of R code can be used for creating the matrices of factor loadings:

```
> matL0 <- mxMatrix(type="Full",
>                   nrow=nv, ncol=2,
>                   free=c(rep(c(TRUE,FALSE),7),
>                          rep(c(FALSE,TRUE),7)),
>                   values=c(rep(c(1,0),7),
>                            rep(c(0,1),7)),
>                   byrow=TRUE,
>                   name="matL0")
> matC1 <- mxMatrix(type="Full",
>                   nrow=nv, ncol=2,
>                   free=c(rep(c(TRUE,FALSE),7),
>                          rep(c(FALSE,TRUE),7)),
>                   values=0,
>                   byrow=TRUE,
>                   name="matC1")
> matC2 <- mxMatrix(type="Full",
>                   nrow=nv, ncol=2,
>                   free=c(rep(c(TRUE,FALSE),7),
>                          rep(c(FALSE,TRUE),7)),
>                   values=0,
>                   byrow=TRUE,
>                   name="matC2")
```

Here, matL0 is a full matrix containing the baseline factor loadings $\boldsymbol{\Lambda}_0$. The first column of the matrix contains the factor loadings of the social inhibition factor, and the second column contains the factor loadings of the negative affectivity factor. By setting free=c(rep(c(TRUE,FALSE),7), rep(c(FALSE,TRUE),7)), values=c(rep(c(1,0),7), rep(c(0,1),7)), and byrow=TRUE, the factor loadings of the social inhibition factor on the first seven items and the factor loadings of the negative affectivity factor on the second seven items are freely estimated (with starting values = 1). For the other factor-indicator combinations, the factor loadings are fixed at zero. Matrices matC1 and matC2 are full matrices containing the direct effects of Male and Age, respectively, on the factor loadings (i.e., nonuniform DIF represented by *c*). These effects are freely estimated in the configural model with starting values of zero by setting free=c(rep(c(TRUE,FALSE),7), rep(c(FALSE, TRUE),7)), values = 0, and byrow=TRUE

The matrices for the residual variances of the indicators can then be specified as symmetric (type="Symm") if there are any nonzero residual covariances, or more simply diagonal (type="Diag") if only residual variances are nonzero (which is the case here):

```
> matE0 <- mxMatrix(type="Diag", nrow=nv,
>                   ncol=nv,
>                   free=TRUE,
>                   values=1,
>                   name="matE0")
> matD1 <- mxMatrix(type="Diag", nrow=nv,
>                   ncol=nv,
>                   free=TRUE,
>                   values=0,
>                   name="matD1")
> matD2 <- mxMatrix(type="Diag", nrow=nv,
>                   ncol=nv,
>                   free=TRUE,
>                   values=0,
>                   name="matD2")
```

where matE0 is a diagonal matrix containing the baseline residual variances $\boldsymbol{\theta}_0$, matD1 is a diagonal matrix containing the effects of Male on the residual variances, and matD2 is a diagonal matrix containing the effects of Age on the residual variances (also represented as *d*). These model parameters are all freely estimated with the free=TRUE argument.

After specifying the matrices of measurement parameters, matrices can be specified for the common-factor variances and correlation. These matrices can be specified as follows:

```
> matP0 <- mxMatrix(type="Symm", nrow=2,
>                   ncol=2,
>                   free=c(FALSE,TRUE,TRUE,
>                          FALSE),
>                   values=c(1,0,0,1),
>                   name="matP0")
> matH1 <- mxMatrix(type="Symm", nrow=2,
>                   ncol=2,
>                   free=c(FALSE,TRUE,TRUE,
>                          FALSE),
>                   values=0,
>                   name="matH1")
> matH2 <- mxMatrix(type="Symm", nrow=2,
>                   ncol=2,
>                   free=c(FALSE,TRUE,TRUE,
>                          FALSE),
>                   values=0,
>                   name="matH2")
```

where matP0 is a symmetric matrix containing the baseline common-factor variances and the baseline correlation[4] between the two common factors. We freely estimate the baseline common-factor correlation and fix the baseline common-factor variances to one in order for the scales of the common factors to be identified. Matrices matH1 and matH2 contain the direct effects of Male and Age, respectively, on the common-factor variances and correlation (also denoted by *h*). In the configural model we set free=c(FALSE,TRUE,TRUE,FALSE) and values=0 in matH1 and matH2, so the direct effects of the background variables on the common-factor correlation are freely estimated, but the direct effects on the common-factor variances are fixed to zero.

The matrices required for the common-factor means can be specified with the following R syntax:

```
> matA0 <- mxMatrix(type="Full", nrow=2,
>                   ncol=1,
>                   free=FALSE,
```

[4] Combining the common-factor variances and correlations in a single matrix may seem odd, but allows us to specify different moderation functions for variance parameters versus correlations (see Bauer, 2017). This is shown in paragraph of Step 3a.

```
>                    values = 0,
>                    name="matA0")
> matG1 <- mxMatrix(type="Full", nrow = 2,
>                    ncol = 1,
>                    free=FALSE,
>                    values = 0,
>                    name="matG1")
> matG2 <- mxMatrix(type="Full", nrow = 2,
>                    ncol = 1,
>                    free=FALSE,
>                    values = 0,
>                    name="matG2")
```

Here, `matA0` is a matrix containing the baseline common-factor means, which are fixed at zero for the origins of the common factors to be identified. The `matG1` and `matG2` matrices contain the direct effects of `Male` and `Age`, respectively, on the common-factor means (also represented by *g*). These direct effects are fixed at zero in the configural model. So, in addition to fixing the baseline common-factor variances and means to one and zero, respectively, the direct effects of the background variables on the common-factor variances and means are also fixed at zero in the configural model. These additional identification constraints are necessary because the configural model includes all possible direct effects of the background variables on the indicators' intercepts, factor loadings and residual variances (analogous to measurement parameters differing across groups in a configural MGCFA model).

Now that all the matrices with baseline parameters and direct effects of the background variables on the parameters have been created, the user can specify the matrices required for the matrix algebra in the next paragraphs. First, to allow for moderating effects, the background variables are modeled as *definition variables* with the following matrices:

```
> matV1 <- mxMatrix(type="Full", nrow = 1,
>                   ncol = 1,
>                   free=FALSE,
>                   labels="data.Male",
>                   name="Male")
> matV2 <- mxMatrix(type="Full", nrow = 1,
>                   ncol = 1,
>                   free=FALSE,
>                   labels="data.Age",
>                   name="Age")
```

The matrices `matV1` and `matV2` contain the observed scores on the background variables `Male` and `Age`, respectively. The observed scores on the background variables stored in the `mxdata1` dataset are referred to in the matrix label as `"data.Male"` and `"data.Age"`. Modeling the background variables as definition variables allows us to let model parameters differ across different levels of `Male` and `Age`.

Then, the matrices for all parameters predicted by `Male` and `Age` are created using the `mxAlgebra()` function. The `mxAlgebra()` function can be used to define a matrix of model parameters as a function of background variables. The first argument of this function, `expression`, should be used for specifying an R expression of one or more `MxMatrix` objects. Most R operators like +, *, and %*%, an general R functions like `mean()`, `log()`, and `exp()` are supported in this argument of the `mxAlgebra()` function. A name for the defined matrix can be assigned with the `name` argument. The matrices of intercepts, factor loadings, and residual variances (respectively) can be specified as

```
> matT <- mxAlgebra(expression = matT0 +
>                   matB1*Male + matB2*Age,
>                   name="matT")
> matL <- mxAlgebra(expression = matL0 +
>                   matC1*Male + matC2*Age,
>                   name="matL")
> matE <- mxAlgebra(expression = matE0*exp
>                   (matD1*Male + matD2*Age),
>                   name="matE")
```

The object `matT` contains a linear moderation function for the indicator intercepts $\boldsymbol{\tau}$ of patient *i* (as in Equation 4):

$$\boldsymbol{\tau}_i = \boldsymbol{\tau}_0 + \boldsymbol{b}_1 \times Male_i + \boldsymbol{b}_2 \times Age_i, \tag{12}$$

and `matL` contains a linear moderation function for the factor loadings $\boldsymbol{\Lambda}$ of patient *i* (as in Equation 5):

$$\boldsymbol{\Lambda}_i = \boldsymbol{\Lambda}_0 + \boldsymbol{C}_1 \times Male_i + \boldsymbol{C}_2 \times Age_i. \tag{13}$$

The object `matE` contains a log-linear function for the residual variances $\boldsymbol{\theta}$ of patient *i* (as in Equation 6):

$$\boldsymbol{\theta}_i = \boldsymbol{\theta}_0 \times \exp(\boldsymbol{d}_1 \times Male_i + \boldsymbol{d}_2 \times Age_i). \tag{14}$$

Similarly, the matrix of common-factor means can be created with the `mxAlgebra()` function:

```
> matA <- mxAlgebra(expression = matA0 +
>                   matG1*Male + matG2*Age,
>                   name="matA")
```

where `matA` contains the common-factor means $\boldsymbol{\alpha}$ being modeled as a linear function of the background variables:

$$\boldsymbol{\alpha}_i = \boldsymbol{\alpha}_0 + \boldsymbol{g}_1 \times Male_i + \boldsymbol{g}_2 \times Age_i. \tag{15}$$

Finally, the covariance matrix for the common factors requires different moderation functions for variances and covariance(s). First, we model the common-factor variances as a log-linear function of `Male` and `Age`:

```
> matVar <- mxAlgebra(expression = matP0*
>                     exp(matH1*Male +
>                     matH2*Age),
>                     name="matVar")
```

The object `matVar` thus contains a matrix with the common-factor variances on the diagonal. For example, the variance of the social inhibition factor is modeled as

$$\psi_{(SISI)i} = \psi_{(SISI)_0} \times \exp(h_{(SISI)1} \times Male_i + h_{(SISI)2} \times Age_i). \tag{16}$$

After we have specified the common-factor variances, we can obtain the common-factor covariance via the common-factor correlation. To respect a correlation's natural bounds between 1 and –1,

we apply a Fisher's $z$ transformation to the correlation, making it a linear function of the background variables (see Equation 10):

```
> matR <- mxAlgebra(expression=
>                   (exp(2*(matP0 + matH1*Male +
>                    matH2*Age)) - 1)/
>                   (exp(2*(matP0 + matH1*Male +
>                    matH2*Age)) + 1),
>                   name="matR")
```

The object `matR` includes a matrix with the common-factor correlation bound between 1 and –1 on the off-diagonal. Before converting factor correlations to covariances, the user must first make a 2 × 2 identity matrix (`matIa`), as well as a 2 × 2 matrix with zeros on the diagonal and ones on the off-diagonal (`matIb`).

```
> matIa <- mxMatrix(type="Diag", nrow = 2, ncol = 2,
>                   free=FALSE,
>                   values = 1,
>                   name="matIa")
> matIb <- mxMatrix(type="Full", nrow = 2, ncol = 2,
>                   free=FALSE,
>                   values=c(0,1,1,0),
>                   name="matIb")
```

We need these matrices to set diagonal and off-diagonal elements of upcoming matrices to zero. Now, the correlation can be converted to a covariance as follows:

```
> matCov <- mxAlgebra(expression=
>                     (matIa*sqrt(matVar))
>                     %*%matR%*%(matIa*sqrt
>                     (matVar)),
>                     name="matCov")
```

Here, we take the square root of the matrix `matVar` to obtain *SD*s on the diagonal, then premultiply this matrix by `matIa` to set all off-diagonal elements to zero. The `matCov` matrix contains the common-factor covariance on the off-diagonal.

Now we can add `matIa*matVar` (i.e., a matrix with common-factor variances on the diagonal and zeros on the off-diagonal) to `matIb*matCov` (i.e., a matrix with zeros on the diagonal and the common-factor covariance on the off-diagonal) to obtain the covariance matrix for the common factors:

```
> matP <- mxAlgebra(expression = matIa*matVar
>                   + matIb*matCov,
>                   name="matP")
```

With all of the matrices specified above (i.e., `matT`, `matL`, `matE`, `matA`, and `matP`), the model-implied moments (means and covariance matrix) of the configural model can be specified, again by using the `mxAlgebra()` function:

```
> matM <- mxAlgebra(expression = matT +
>                   t(matL)*matA,
>                   name="matM")
> matC <- mxAlgebra(expression = matL
>                   %*% matP%*%t(matL) + matE,
>                   name="matC")
```

The `matM` matrix contains the model-implied means, and the `matC` matrix contains the model-implied variances and covariances of the indicators.

In order to fit the configural model with the specified model-implied matrices, the user needs to specify the model expectation and fit function:

```
> expF <- mxExpectationNormal
>         (covariance="matC",
>         means = "matM",
>         dimnames=manVars)
> fitF <- mxFitFunctionML()
```

The expectation function stored in `expF` defines the way in which the model expectations are calculated. The `mxExpectationNormal()` function uses the algebra defined in `matC` and `matM` to obtain the model-implied variances, covariances, and means of the indicators under multivariate normality. The `dimnames` argument of the function takes the character vector `manVars` containing the names of the indicators. The `mxFitFunctionML()` function stored in `fitF` is used to indicate that the free parameters of the configural model should be estimated using full-information maximum likelihood. Alternatively, a user-defined fit function can be treated as an `mxFitFunction` by using the `mxFitFunctionR()` function.

All of the separate objects for each part of the configural model can now be combined using the `mxModel()` function. The `model` argument of this function can be used to specify a name for the new model. The following arguments are a number of `MxMatrix` and `MxAlgebra` objects, as well as the expectation function, fit function, and `MxData` objects. All of these objects can be added to the model as follows:

```
> modConfig <- mxModel(model="Configural",
>                      matT, matT0, matB1,
>                      matB2,matL, matL0,
>                      matC1, matC2, matE,
>                      matE0, matD1, matD2,
>                      matP, matP0, matH1,
>                      matH2, matA, matA0,
>                      matG1, matG2, matIa,
>                      matIb, matV1, matV2,
>                      matVar, matR, matCov,
>                      matM, matC, expF,
>                      fitF, mxdata1)
```

The object `modConfig` now includes the data, model matrices, expectation function, and fit function. These objects are all the required elements to optimize the free parameters in the model. The model can be fitted to the data using the `mxRun()` function. This function sends the `MxModel` object specified in the first argument to the optimizer and returns the optimized model. Additional information on the parameters estimates can be requested by including arguments like `intervals = TRUE` for confidence intervals. The configural model can be fitted to the DS14 data as follows:

```
> fitConfig <- mxRun(modConfig)
```

The output can be printed using the `summary()` function. The summary output of the model contains the estimates of all free parameters, their standard errors, and model statistics including the

number of parameters and goodness-of-fit reported in units of minus two times the log-likelihood (−2lnL). In Step 3c, this fit statistic will be compared with the fit of the scalar model (Step 3b).

### *Step 3b: Specify and Fit the Scalar Model*

In this step, we show how to specify and fit the scalar model in which all direct effects of Male and Age on the indicators' intercepts and factor loadings are fixed at zero. The user should first respecify matrices matB1, matB2, matC1, and matC2:

```
> matB1 <- mxMatrix(type="Full", nrow = 1,
>                   ncol=nv,
>                   free=FALSE,
>                   values = 0,
>                   name="matB1")
> matB2 <- mxMatrix(type="Full", nrow = 1,
>                   ncol=nv,
>                   free=FALSE,
>                   values = 0,
>                   name="matB2")
> matC1 <- mxMatrix(type="Full", nrow=nv,
>                   ncol = 2,
>                   free=FALSE,
>                   values = 0,
>                   name="matC1")
> matC2 <- mxMatrix(type="Full", nrow=nv,
>                   ncol = 2,
>                   free=FALSE,
>                   values = 0,
>                   name="matC2")
```

These new matrices for the direct effects on the intercepts and factor loadings indicate that none of the directs effect of the background variables on the indicator's intercept and factor loading should be freely estimated by using the free=FALSE argument in each of these matrices.

Next, we release the additional identification constraints that were necessary for the configural model in Step 3a. That is, in the scalar model we freely estimate the direct effects of Male and Age on the common-factor means and variances by respecifying matrices matH1, matH2, matG1, and matG2:

```
> matH1 <- mxMatrix(type="Symm", nrow = 2,
>                   ncol = 2,
>                   free=TRUE,
>                   values = 0,
>                   name="matH1")
> matH2 <- mxMatrix(type="Symm", nrow = 2,
>                   ncol = 2,
>                   free=TRUE,
>                   values = 0,
>                   name="matH2")
> matG1 <- mxMatrix(type="Full", nrow = 2,
>                   ncol = 1,
>                   free=TRUE,
>                   values = 0,
>                   name="matG1")
> matG2 <- mxMatrix(type="Full", nrow = 2,
>                   ncol = 1,
>                   free=TRUE,
>                   values = 0,
>                   name="matG2")
```

These matrices now indicate that the common-factor variances and means are allowed to differ across all values of Male and Age. All other elements of the scalar model are similar to the configural model specified in Step 3a. So, after respecifying objects matB1, matB2, matC1, matC2, matH1, matH2, matG1, and matG2 we can combine the elements required to run the scalar model and fit the model to the data:

```
> modScalar <- mxModel(model="Scalar",
>                      matT, matT0, matB1,
>                      matB2, matL, matL0,
>                      matC1, matC2,
>                      matE, matE0, matD1,
>                      matD2, matP, matP0,
>                      matH1, matH2, matA,
>                      matA0, matG1, matG2,
>                      matIa, matIb, matV1,
>                      matV2,matVar, matR,
>                      matCov, matM, matC,
>                      expF, fitF, mxdata1)
> fitScalar <- mxRun(modScalar)
```

Again, the summary() function can be used to obtain the model fit and parameter estimates.

### *Step 3c: Conduct Likelihood-Ratio Test*

Now that both the configural and scalar model have been fitted to the data, a likelihood ratio test (LRT) can be performed using the mxCompare() function:

```
> miTest <- mxCompare(fitConfig, fitScalar)
```

Using $\alpha = .05$ as significance level, the output shows that the constraints on the intercepts and factor loadings significantly deteriorate model fit, $\Delta\chi^2(48) = 112.52$, $p < .001$. This indicates that the $H_0$ of full scalar-and-metric invariance can be rejected. In the following steps, we will illustrate how follow-up tests can be performed to evaluate which indicators function differently with respect to the background variables.

## Step 4: Select Anchor Indicators

In the previous step, we have rejected the $H_0$ of full scalar-and-metric invariance with respect to Male and Age. Partial invariance can be established by detecting which specific indicators exhibit (non)uniform DIF. Each indicator can be tested individually for DIF while holding a subset of other indicators invariant across the background variables. These latter indicators are also called *anchor indicators* and are used to link the metric of the common factors across the background variables. When anchor indicators are not known a priori, they can be explicitly selected using an anchor-selection strategy. In this step of the tutorial, we show how to apply the rank-based strategy (Woods, 2009) for the selection of anchor indicators. This is an easily implemented selection strategy in which a limited number of indicators that show the weakest evidence of DIF are selected as anchor indicators. More complicated empirical methods for selecting anchors can slightly improve accuracy (Kopf et al., 2015a, 2015b), and regularized estimation can avoid the need for anchors altogether (Bauer et al., 2020). The potential danger of selecting anchor indicators is that one or more indicators with

DIF may be selected as anchors, which can cause problems such as biased parameter estimates and an overestimation of the amount of DIF (W.-C. Wang, 2004). However, the risk of bias can be minimal if positive and negative DIF are relatively balanced. When DIF is unbalanced, the risk of such a contamination of the subset of anchor indicators can be decreased by selecting a relatively small anchor set. Accordingly, Woods' (2009) recommended to select approximately 20% of indicators to serve as anchor indicators. We follow this recommendation by selecting two out of seven indicators per common factor to serve as anchor indicators across both background variables.[5] One could also argue to select anchor indicators for each background variable separately, but to keep the following steps of the tutorial as concise as possible we demonstrate how to select the same anchor indicators for `Male` and `Age`.

### *Step 4a: Specify and Fit All-but-One Models*

The first step of the rank-based strategy is to fit a less-constrained *all-but-one model* for each indicator. In an indicator's all-but-one model, only that indicator's intercept and factor loading are predicted by the background variables (i.e., all but one of the indicators are assumed to be scalar-and-metric invariant). For example, the all-but-one model for Indicator 1 (`si1`) includes freely estimated direct effects of `Male` and `Age` on the intercept and factor loading of Indicator 1, and the direct effects of the background variables on all other indicators' intercepts and factor loadings are fixed at zero. So, almost all of the matrices specified in Step 3a can be used for these models, except for matrices `matB1`, `matB2`, `matC1`, and `matC2`.

In order to efficiently execute this step, we specify and fit an all-but-one model for each indicator in a `for` loop. First, we create an empty list to which we can add each model's output:

```
> fitAbo <- list()
```

Next, we run the following `for` loop:

```
> for (i in 1:nv){
>     freeparT <- matrix(data=FALSE, nrow = 1,
>                        ncol=nv)
>     freeparT[i] <- TRUE
>     freeparL <- matrix(data=FALSE, nrow=nv,
>                        ncol = 2)
>     freeparL[i, ifelse(i < 8, yes = 1, no = 2)]
>                        <- TRUE
>     matB1 <- mxMatrix(type="Full", nrow = 1,
>                       ncol=nv,
>                       free=freeparT,
>                       values = 0,
>                       name="matB1")
>     matB2 <- mxMatrix(type="Full", nrow = 1,
>                       ncol=nv,
>                       free=freeparT,
>                       values = 0,
>                       name="matB2")
>     matC1 <- mxMatrix(type="Full", nrow=nv,
>                       ncol = 2,
>                       free=freeparL,
>                       values = 0,
>                       byrow=TRUE,
>                       name="matC1")
>     matC2 <- mxMatrix(type="Full", nrow=nv,
>                       ncol = 2,
>                       free=freeparL,
>                       values = 0,
>                       byrow=TRUE,
>                       name="matC2")
>     modAbo <- mxModel(model=paste0 ("All_but_", i),
>                       matT, matT0, matB1, matB2,
>                       matL, matL0, matC1, matC2,
>                       matE, matE0, matD1, matD2,
>                       matP, matP0, matH1, matH2,
>                       matA, matA0, matG1, matG2,
>                       matIa, matIb, matV1, matV2,
>                       matVar, matR, matCov, matM, matC,
>                       expF, fitF, mxdata1)
>     fitAbo[[i]] <- mxRun(modAbo)
>}
```

In this `for` loop, the syntax will run for each of the 14 indicators represented by `i`. First, we specify which elements in `matB1`, `matB2`, `matC1`, and `matC2` should be freely estimated by creating matrices with true and false entries, named `freeparT` and `freeparL`. These matrices are used for the `free` argument of the `mxMatrix()` function to indicate that the direct effects of the background variables on indicator `i`'s intercept and factor loading should be freely estimated. The effects of the background variables on all intercepts and factor loadings of indicators other than `i` are fixed to zero (implying no DIF). After respecifying the matrices `matB1`, `matB2`, `matC1`, and `matC2`, all elements required to fit the all-but-one model of indicator i are combined using the `mxModel()` function. The model is then optimized using `mxRun()` and added to the ith component of the list `fitAbo`.

### *Step 4b: Conduct Likelihood-Ratio Tests and Select Anchors*

After specifying and fitting the less-constrained all-but-one models, each one's fit can be compared with the fit of the constrained scalar-invariance model (`fitScalar`). A comparison of the fit of these models with a LRT indicates whether additionally fixing the current indicator's intercept and factor loading to be unaffected by the background variables leads to a significantly worse model fit. Note that these LRTs should not be trusted as tests of DIF because unmodeled DIF biases other parameters, inflating Type I error rates for DIF-free indicators; however, these tests can serve as a reliable empirical basis for selecting anchors (Kolbe & Jorgensen, 2019; Kopf et al., 2015a; Woods, 2009). Because `fitAbo` is a list of fitted models, each of them will be compared to `fitScalar`, and we store the LRT results in a readable table `anchorOut`:

```
> anchorTest <- mxCompare(fitAbo, fitScalar)
> anchorOut <- data.frame(Name = paste0
```

[5] Note that if only one anchor indicator per common factor is used for identification, the model would simply be a configural model, equivalent to the one specified in Step 3a. Such a model is incapable of distinguishing between differences in common-factor means and variances from differences in the reference-indicator's intercept and loading across the background variables. That is why at least two anchor indicators are required per common factor for differences in common-factor means and variances to be attributed to differences in the common-factor distribution, rather than being due to differences in a single indicator.

```
>                    ("Indicator", 1:nv),
>                X2 = anchorTest$diffLL
>                    [seq(2,28,2)],
>                df = anchorTest$diffdf
>                    [seq(2,28,2)],
>                 p = anchorTest$p
>                    [seq(2,28,2)])
```

Differences in –2lnL values from the scalar-invariance and all-but-one models follow a $\chi^2$ distribution with $df = 4$. The results stored in `piOut` are presented in Table 1.

Indicators 1–7 of social inhibition and 8–14 of negative affectivity can now be ranked in ascending order based on their LRT statistics:

```
> anchorOut[order(anchorOut$X2[1:7]),]
> anchorOut[7+order(anchorOut$X2[8:14]),]
```

The smaller the test statistic, the weaker the evidence of DIF. So, for each common factor, the two indicators with the smallest test statistics are selected as anchor indicators:

```
> anchors1  <- c(5, 6)
> anchors2  <- c(8, 9)
```

In this dataset, Indicators 5 and 6 are selected as anchor indicators for social inhibition, and Indicators 8 and 9 are selected as anchor indicators for negative affectivity. The indicator indices are stored in `anchors1` and `anchors2` to use as anchors in Step 5, when we test all other *studied indicators* (i.e., Indicators 1, 2, 3, 4, 7, 10, 11, 12, 13, 14) for DIF.

## Step 5: Assess Partial Invariance

Previously in Step 3, we found evidence against full scalar-and-metric invariance with respect to the background variables `Male` and `Age`. We can now perform follow-up tests in order to evaluate partial scalar-and-metric invariance. We will show how to use MNLFA to test the $H_0$ of invariance for each indicator by comparing the fit of a less-constrained *anchors-only model* to several more-constrained *anchors-plus-one models*. In the anchors-only model, the direct effects of the background variables on all studied indicators' intercepts and factor loadings are freely estimated to allow for (non)uniform DIF, so only the anchor indicators have scalar- and metric-invariance constraints. For each studied indicator, its anchors-plus-one model additionally constrains that indicator to be invariant by removing the background variables' (non) uniform DIF estimates. Because background variables continue to affect remaining studied indicators, parameter estimates in these anchors-plus-one models are not biased by DIF (unless selected anchors have DIF, which is a small risk; Kolbe & Jorgensen, 2019; Kopf et al., 2015b; Woods, 2009).

**Table 1**
*Likelihood-Ratio Tests for the Purpose of Selecting Anchor Indicators*

| Indicator | Name | $\Delta\chi^2(4)$ | $p$ | Rank |
|---|---|---|---|---|
| 1 | Si1 | 16.36 | .003 | 6 |
| 2 | Si3 | 8.78 | .067 | 5 |
| 3 | Si6 | 5.44 | .245 | 3 |
| 4 | Si8 | 7.42 | .115 | 4 |
| 5 | Si10 | 4.30 | .367 | 2 |
| 6 | Si11 | 3.05 | .550 | 1 |
| 7 | Si14 | 16.72 | .002 | 7 |
| 8 | Na2 | 2.43 | .658 | 2 |
| 9 | Na4 | 1.15 | .886 | 1 |
| 10 | Na5 | 11.02 | .026 | 5 |
| 11 | Na7 | 5.36 | .252 | 4 |
| 12 | Na9 | 3.90 | .420 | 3 |
| 13 | Na12 | 22.54 | .000 | 7 |
| 14 | Na13 | 19.38 | .001 | 6 |

*Note*. The rank score is based on the LRT statistic (i.e., $\Delta\chi^2(4)$) of each indicator per common factor. The indicators are ranked in ascending order, that is, a higher rank score indicates a smaller LRT statistic and thus a weaker evidence of DIF.

### *Step 5a: Specify and Fit the Anchors-Only Model*

We first create an object containing the studied indicators, by removing anchors:

```
> testIn <- c(1:nv)[-c(anchors1, anchors2)]
```

Then, we create two matrices that indicate which direct effects of the background variables are freely estimated in the anchors-only model:

```
> freeparT <- matrix(TRUE, nrow = 1, ncol = 14)
> freeparT[1, c(anchors1, anchors2)] <- FALSE
> freeparL <- matrix(c(rep(c(TRUE,FALSE),7),
>                      rep(c(FALSE,TRUE),7)),
>                    nrow=nv, ncol = 2, byrow = TRUE)
> freeparL[anchors1, 1] <- FALSE
> freeparL[anchors2, 2] <- FALSE
```

After running these lines of syntax, `freeparT` and `freeparL` are matrices with `TRUE` and `FALSE` entries indicating which intercepts and factor loadings, respectively, are allowed to differ as a function of the background variables.

In the anchors-only model, all studied indicators' intercepts and factor loadings are now allowed to differ as a function of the background variables. In order to specify the anchors-only model, the user should indicate which elements in `MxMatrix` objects `matB1`, `matB2`, `matC1`, and `matC2` can be freely estimated:

```
> matB1 <- mxMatrix(type="Full", nrow = 1, ncol=nv,
>                   free=freeparT,
>                   values = 0,
>                   name="matB1")
> matB2 <- mxMatrix(type="Full", nrow = 1, ncol=nv,
>                   free=freeparT,
>                   values = 0,
>                   name="matB2")
> matC1 <- mxMatrix(type="Full", nrow=nv, ncol = 2,
>                   free=freeparL,
>                   values = 0,
>                   name="matC1")
> matC2 <- mxMatrix(type="Full", nrow=nv, ncol = 2,
>                   free=freeparL,
>                   values = 0,
>                   name="matC2")
```

All other elements of the anchors-only model are similar to the all-but-one models specified in Step 4a. So after respecifying

matB1, matB2, matC1, and matC2, the anchors-only model can be fitted to the DS14 data:

```
> modAnchors <- mxModel(model="AnchorsOnly",
>                       matT, matT0, matB1,
>                       matB2,matL, matL0,
>                       matC1, matC2, matE,
>                       matE0, matD1, matD2,
>                       matP, matP0, matH1,
>                       matH2, matA, matA0,
>                       matG1, matG2, matIa,
>                       matIb, matV1, matV2,
>                       matVar, matR, matCov,
>                       matM, matC,
>                       expF, fitF, mxdata1)
> fitAnchors <- mxRun(modAnchors)
```

The object fitAnchors contains the model fit and parameter estimates of the unconstrained model, which can be printed using the summary() function.

### *Step 5b: Specify and Fit Anchors-Plus-One Models*

In each anchors-plus-one model, the studied indicator is additionally constraint to exhibit no DIF. That is, all intercepts and factor loadings are allowed to differ as a function of the background variables, except for the current studied indicator and the anchors. First, an empty list can be created for the output of all constrained models:

```
> fitApo <- list()
```

The anchors-plus-one model for each studied indicator can be specified and fit within this for() loop:

```
> for (i in testIn){
>     freeparTa <- freeparT
>     freeparLa <- freeparL
>     freeparTa[i] <- FALSE
>     freeparLa[i, ifelse(i < 8, yes = 1, no = 2)]
>                <- FALSE
>     matB1 <- mxMatrix(type="Full", nrow = 1, ncol=nv,
>                       free=freeparTa,
>                       values = 0,
>                       name="matB1")
>     matB2 <- mxMatrix(type="Full", nrow = 1, ncol=nv,
>                       free=freeparTa,
>                       values = 0,
>                      name="matB2")
>     matC1 <- mxMatrix(type="Full", nrow=nv, ncol = 2,
>                       free=freeparLa,
>                       values = 0,
>                       name="matC1")
>     matC2 <- mxMatrix(type="Full", nrow=nv, ncol = 2,
>                       free=freeparLa,
>                       values = 0,
>                       name="matC2")
>    modApo <- mxModel(model=paste0("Anchors_plus_", i),
>                      matT, matT0, matB1, matB2,
>                      matL, matL0, matC1, matC2,
>                      matE, matE0, matD1, matD2,
>                      matP, matP0, matH1, matH2,
>                      matA, matA0, matG1, matG2,
>                      matIa, matIb, matV1, matV2,
>                      matVar, matR, matCov, matM, matC,
>                      expF, fitF, mxdata1)
>     fitApo[[i]] <- mxRun(modApo)
>}
```

Therefore, for each studied indicator, the matrices of freely estimated (non)uniform DIF (freeparT and freeparL) are copied in the first two lines of the for() loop, in order to additionally fix the DIF parameters of the current studied indicator i to zero in the following two lines. After completing the for() loop, the object fitApo is a list containing the fit and parameter estimates of each studied indicators' anchors-plus-one model.

### *Step 5c: Conduct Likelihood-Ratio Tests*

Partial scalar-and-metric invariance can now be assessed by performing a LRT for all studied indicators using the mxCompare() function:

```
> piTest <- mxCompare(fitAnchors, fitApo)
> piOut <- data.frame(Name = paste0
>                     ("Indicator", testIn),
>                      X2 = piTest$diffLL[2:11],
>                      df = piTest$diffdf[2:11],
>                      p = piTest$p[2:11],
>                      p.bon = p.adjust(p=piTest$
>                      p[2:11], method="bonferroni"),
>                      p.BH = p.adjust(p=piTest$
>                      p[2:11], method="BH"))
```

The object piOut contains the LRT statistic, *df*, and *p* value for each studied indicator, presented in Table 2. To account for multiple testing, we also included Bonferroni-adjusted *p* values to control the familywise type I error rate, as well as more-powerful Benjamini–Hochberg adjustments to maintain a false discovery rate no larger than the $\alpha$ level. Without accounting for multiple testing, the LRTs indicate that constraining the intercepts and factor loadings of Indicators 1, 2, 7, and 13 to be unaffected by Male and Age leads to a significantly worse model fit. Using $\alpha = .05$ as significance level, the $H_0$ of measurement invariance with respect to Male and Age for these indicators can thus be rejected. However, other conclusions can be made when accounting for multiple testing. The Bonferroni-adjusted *p* values indicate that only Indicator 13 significantly violates scalar-and-metric invariance, whereas the Benjamini–Hochberg-adjusted *p* values additionally indicate a significant violation of Indicator 7. Follow-up Wald tests of specific (non)uniform DIF can be conducted by consulting the Wald *z* statistics in the summary() output of the models with significant DIF. This may be warranted if more information about the nature of DIF is desired (e.g., to attempt revising indicators to remove such DIF). Our Open Science Framework project https://osf.io/6cyxt/ also includes R code to inspect tracelines of the DIF indicators which may help with interpreting the DIF effects. In addition, R code for plots of moderated common-factor means, variances, and correlations can be found here along with image files for the plots themselves.

## Final Model: Comparison With Mplus

In this section of the tutorial, we fit the final partial-invariance model to the DS14 data, using both OpenMx (Neale et al., 2016)

**Table 2**
*Likelihood-Ratio Tests for Assessing Partial Invariance*

| Indicator | Name | $\Delta\chi^2(4)$ | $p$ | $P_{bon}$ | $p_{ben–hoc}$ |
|---|---|---|---|---|---|
| 1 | Si1 | 10.67 | **.031** | .305 | .089 |
| 2 | Si3 | 10.30 | **.036** | .357 | .089 |
| 3 | Si6 | 5.69 | .224 | 1.000 | .287 |
| 4 | Si8 | 5.61 | .230 | 1.000 | .287 |
| 7 | Si14 | 14.44 | **.006** | .060 | **.030** |
| 10 | Na5 | 8.50 | .075 | .748 | .150 |
| 11 | Na7 | 1.64 | .802 | 1.000 | .802 |
| 12 | Na9 | 3.03 | .554 | 1.000 | .615 |
| 13 | Na12 | 15.27 | **.004** | **.042** | **.030** |
| 14 | Na13 | 5.91 | .206 | 1.000 | .287 |

*Note*. Indicators 5 and 6 are the anchors for social inhibition and Indicators 8 and 9 are the anchors for negative affectivity. The bold cells indicate significant (non)uniform DIF based on the original $p$ value, the Bonferroni-adjusted $p$ value denoted $p_{bon}$, and the Benjamini–Hochberg-adjusted $p$ value denoted $p_{ben–hoc}$.



and Mplus (Muthén & Muthén, 2012). The purpose of this section is to show that the parameter estimates obtained by these two software packages are identical. In the final partial-invariance model, we assume scalar and metric invariance of all indicators except for Indicators 7 and 13. These indicators functioned differently with respect to the background variables, which is taken into account in the final partial-invariance model by freely estimating the effects of `Male` and `Age` on the intercept and factor loading of this indicator.

First, we can specify which indicators are scalar and metric invariant:

```
> finalIn <- c(1,2,3,4,5,6,8,9,10,11,12,14)
```

We can then use this `finalIn` object to indicate which effects of the background variables `Male` and `Age` on the intercepts and factor loadings should be freely estimated:

```
> freeparTb <- freeparT
> freeparLb <- freeparL
> for(i in finalIn){
>   freeparTb[1, i] <- FALSE
>   freeparLb[i, ifelse(i < 8, yes = 1, no = 2)]
>             <- FALSE
>}
```

The matrices `freeparTb` and `freeparLb` indicate which DIF parameters should be estimated in the partial-invariance model. The matrices `matB1`, `matB2`, `matC1`, and `matC2` can now be respecified:

```
> matB1 <- mxMatrix(type="Full", nrow = 1, ncol=nv,
>                   free=freeparTb,
>                   values = 0,
>                   name="matB1")
> matB2 <- mxMatrix(type="Full", nrow = 1, ncol=nv,
>                   free=freeparTb,
>                   values = 0,
>                   name="matB2")
> matC1 <- mxMatrix(type="Full", nrow=nv, ncol = 2,
>                   free=freeparLb,
>                   values = 0,
>                   name="matL1")
> matC2 <- mxMatrix(type="Full", nrow=nv, ncol = 2,
>                   free=freeparLb,
>                   values = 0,
>                   name="matC2")
```

and added to a new `MxModel` object `modOpenmxPartial`, which can then be fitted to data using the `mxRun()` function:

```
> modOpenmxPartial <- mxModel(model=
>                             "PartialInvariance",
>                             matT, matT0, matB1,
>                             matB2, matL, matL0,
>                             matC1, matC2, matE,
>                             matE0, matD1, matD2,
>                             matP, matP0, matH1,
>                             matH2, matA, matA0,
>                             matG1, matG2, matIa,
>                             matIb, matV1, matV2,
>                             matVar, matR, matCov,
>                             matM, matC, expF,
>                             fitF, mxdata1)
> fitOpenmxPartial <- mxRun(modOpenmxPartial)
```

Again, the `summary()` function can be used to evaluate the fit of this model. The parameter estimates of this model are shown in the summary output and can be extracted with `summary(fitOpenmxPartial)$parameters`.

The same partial-invariance model can also be fitted to the data in M*plus* via the R package `MplusAutomation` (Hallquist & Wiley, 2018). Note that the R syntax below can only be executed if Mplus is installed on the user's computer. The first step of fitting the partial-invariance model in Mplus is to create a folder in which the empirical data and models can be stored:

```
> pathfix <- "~/FinalModel"
> dir.create(pathfix)
```

The DS14 data can then be saved in this directory using the prepareMplusData() function:

```
> prepareMplusData(df=DS14, filename=
>                    paste0(pathfix,
>                    "/DS14dat.dat"))
```

The DS14 data can then be saved in this directory using the prepareMplusData() function:

```
> prepareMplusData(df=DS14, filename=
    paste0(pathfix, "/DS14dat.dat"))
```

Now that the "/FinalModel" folder contains the empirical data, the partial-invariance model can be specified and added to the directory:

```
> modMplusPartial <- '
>     DATA:
>     FILE = "DS14dat.dat";
>
>     VARIABLE:
>     NAMES = Male Age x1 x2 x3 x4 x5 x6 x7 x8
>           x9 x10 x11 x12 x13 x14;
>     CONSTRAINT = Male Age;
>     MISSING = .;
>
>     ANALYSIS:
>     estimator is ML;
>
>     MODEL:
>     SI by x1-x7* (l1-l7);
>     NA by x8-x14* (l8-l14);
>     SI (SIvar);
>     NA (NAvar);
>     SI with NA (cov);
>     [SI] (SImean);
>     [NA] (NAmean);
>     [x1-x14] (t1-t14);
>     x1-x14 (e1-e14);
>     Male@1;
>     Age@1;
>     [Male@0];
>     [Age@0];
>
>     MODEL CONSTRAINT:
>     NEW (
>     h1_SI h1_NA g1_SI g1_NA
>     h2_SI h2_NA g2_SI g2_NA
>     p0 h1_cov h2_cov
>     t7_0 b7_1 b7_2 t13_0 b13_1 b13_2
>     l7_0 c7_1 c7_2 l13_0 c13_1 c13_2
>     e1_0 e2_0 e3_0 e4_0 e5_0 e6_0 e7_0
>     e8_0 e9_0 e10_0 e11_0 e12_0 e13_0 e14_0
>     d1_1 d2_1 d3_1 d4_1 d5_1 d6_1 d7_1
>     d8_1 d9_1 d10_1 d11_1 d12_1 d13_1 d14_1
>     d1_2 d2_2 d3_2 d4_2 d5_2 d6_2 d7_2 d8_2
>     d9_2 d10_2 d11_2 d12_2 d13_2 d14_2);
>     SIvar = 1 * EXP(h1_SI*Male + h2_SI*Age);
>     NAvar = 1 * EXP(h1_NA*Male + h2_NA*Age);
>     SImean = 0 + g1_SI*Male + g2_SI*Age;
>     NAmean = 0 + g1_NA*Male + g2_NA*Age;
>     cov = SQRT(EXP(h1_SI*Male + h2_SI*Age))*
>         SQRT(EXP(h1_NA*Male + h2_NA*Age))*
>         (EXP(2*(p0 + h1_cov*Male +
>             h2_cov*Age))-1)/
>         (EXP(2*(p0 + h1_cov*Male +
>             h2_cov*Age))+1);
>     e1 = e1_0 * EXP(d1_1*Male + d1_2*Age);
>     e2 = e2_0 * EXP(d2_1*Male + d2_2*Age);
>     e3 = e3_0 * EXP(d3_1*Male + d3_2*Age);
>     e4 = e4_0 * EXP(d4_1*Male + d4_2*Age);
>     e5 = e5_0 * EXP(d5_1*Male + d5_2*Age);
>     e6 = e6_0 * EXP(d6_1*Male + d6_2*Age);
>     e7 = e7_0 * EXP(d7_1*Male + d7_2*Age);
>     e8 = e8_0 * EXP(d8_1*Male + d8_2*Age);
>     e9 = e9_0 * EXP(d9_1*Male + d9_2*Age);
>     e10 = e10_0 * EXP(d10_1*Male + d10_2*Age);
>     e11 = e11_0 * EXP(d11_1*Male + d11_2*Age);
>     e12 = e12_0 * EXP(d12_1*Male + d12_2*Age);
>     e13 = e13_0 * EXP(d13_1*Male + d13_2*Age);
>     e14 = e14_0 * EXP(d14_1*Male + d14_2*Age);
>     t7 = t7_0 + b7_1*Male + b7_2*Age;
>     l7 = l7_0 + c7_1*Male + c7_2*Age;
>     t13 = t13_0 + b13_1*Male + b13_2*Age;
>     l13 = l13_0 + c13_1*Male + c13_2*Age;'
> cat(modMplusPartial, file = paste0
>     (pathfix, "/modPartial.inp", sep = ""))
```

For a more detailed explanation of the Mplus syntax, see Bauer's (2017) supplementary materials. For the sake of simplifying comparison of results, we use MplusAutomation so that results can be imported into R, although R is not necessary to use Mplus for MNLFA estimation. If one prefers to run the models in Mplus directly instead of indirectly via MplusAutomation, the Mplus script can be found on our Open Science Framework project https://osf.io/6cyxt/.

The "/modPartial.inp" file can be run and the corresponding results can be imported into R as follows:

```
> runModels(pathfix)
> fitMplusPartial <- readModels(pathfix)
```

The parameter estimates of this model stored in fitMplusPartial$parameters are identical to those obtained by OpenMx, as seen with summary(fitOpenmxPartial)$parameters. This provides a valuable cross-validation that the model is specified equivalently in both software packages and that both optimizers converge on the same full-information maximum likelihood estimates. Further research is needed to cross-validate across a wider variety of SEMs (e.g., categorical outcomes; Bauer, 2017), but the current results imply that any researcher can utilize MNLFA with the freely available OpenMx package, even if they do not have access to commercial software such as Mplus or SAS.

## Extensions

### Nonlinear Effects Among Background Variables

One may want to add quadratic effects of a background variable, or in the case of MNLFA with multiple background variables, it may be desirable to account for the interactions between the background variables in their effects on factor-model parameters. Incorporating quadratic effects or interactions between background variables is straightforward. We simply need to calculate the quadratic or interaction variable in R and treat it as an additional moderator in OpenMx. For instance, if we are interested in the interaction between Age and Male in the real data example above, we can use:

```
> DS14$Int = DS14$Age * DS14$Male
> mxdata_int <- mxData(observed=DS14,
```

```
>                          type="raw")
> matV2 <- mxMatrix(type="Full", nrow = 1, ncol = 1,
>                   free=FALSE,
>                   labels="data.Int",
>                   name="Int")
```

Now we can specify effects of this additional moderator along with the other effects on factor-model parameters—for example, the factor loadings:

```
> matB3 <- mxMatrix(type="Full", nrow = 1, ncol=nv,
>                   free=freeparT,
>                   values = 0,
>                   name="matB3")
> matT <- mxAlgebra(expression = matT0 +
>                   matB1*Male + matB2*Age +
>                   matB3*Int,
>                   name="matT")
```

That is, an additional parameter matrix, `matB3`, specifies the `Age × Male` interaction effects on factor loadings. We add these effects to the `expression` for the moderation of the factor loadings. For all other moderated parameters (intercepts, residual variances, factor means, and factor variances), the procedure is the same: An extra parameter matrix needs to be specified, and the interaction between the background variables needs to be added to the moderation function. Likewise, the quadratic effect of `Age` could be added by calculating a new variable that is the square of `Age`, then specifying additional parameter matrices for its effects.

## Ordinal Data

If the indicators in the MNLFA model are ordinal, it may be advisable to explicitly treat them as ordinal (particularly with $< 5$ categories) to prevent detection of spurious nonlinear effects. To do so, we must first specify that the data are ordinal. This can be done using the `mxFactor()` function:

```
> DS14[,3:16] <- mxFactor(DS14[,3:16], levels = 0:4)
> mxdata_ord <- mxData(observed=DS14, type="raw")
> nc <- 5
```

Thus, we specified the indicators (i.e., the data in Columns 3 to 16 of the `DS14` matrix) to be ordinal with levels 0 to 4. Additionally, we assigned the data to object `mxdata_ord` and we specified an object (`nc`) to denote the number of categories per variable (5), which we use below to specify the model.

Next, to enable MNLFA of ordered categorical data, a discrete factor model is adopted for the data (for a formal description of such models, see Muthén, 1984; Takane & De Leeuw, 1987; Wirth & Edwards, 2007). In the discrete factor model, each observed ordinal indicator is assumed to have a corresponding normally distributed latent response variable (LRV) that is discretized by thresholds to yield the ordered categories. Therefore, to extend the code for the unconstrained model above, we specify a matrix containing the threshold parameters. If the ordinal indicators contain $k$ categories, there are $k - 1$ threshold parameters per indicator. We therefore need a $(k - 1) \times p$ matrix in which rows represent thresholds and columns represent indicators:

```
> matThres <- mxMatrix(type="Full", nrow=nc-1, ncol=nv,
>                      free=TRUE,
>                      values=rep(seq(-2,2,
>                                 length.out=nc-1),
>                                 times=nv),
>                      byrow=FALSE,
>                      name="matThres")
```

Recall `nc` denotes a constant[6] number of categories per indicator (see above). The values for the thresholds can be negative, but should always be increasing for a given indicator. Therefore, as starting values, we specified increasing, equally spaced values between –2 and 2.

In the discrete factor model, the origin and scale of each LRV must be set, just as for latent common factors (Wu & Estabrook, 2016). In the code below, we fix the baseline intercepts to zero and baseline residual variances to one for all ordinal indicators, allowing us to freely estimate all[7] their thresholds.

```
> matT0 <- mxMatrix(type="Full", nrow = 1, ncol=nv,
>                   free=FALSE,
>                   values = 0,
>                   name="matT0")
> matE0 <- mxMatrix(type="Diag", nrow=nv, ncol=nv,
>                   free=FALSE,
>                   values = 1,
>                   name="matE0")
```

Using the same `matThres` across all values of the background variables implies invariance of thresholds across the background variables, in which case the configuration of background-variable effects on all other model parameters need not change from those shown in Steps 4 and 5. We discuss this assumption of equal thresholds after fitting the model.

Then we need to respecify the expectation function to indicate the presence of thresholds in the model:

```
> expF <- mxExpectationNormal(covariance="matC",
>                             means = "matM",
>                             thresholds="matThres",
>                             dimnames=manVars)
```

Finally, we can combine all objects into an `OpenMx` object and add the restriction that the thresholds are strictly increasing for a given indicator:

```
> modUn_thres <- mxModel(model="ThresholdInvariance",
>                        matT, matT0, matB1, matB2,
```

[6] In the case of an unequal number of categories across indicators, `nc` should be set to the maximum number of thresholds across all indicators. Thresholds that do not exist can be dropped from the model by specifying the `values=` argument to be `NA` (missing) and the `free=` argument as `FALSE` for nonexisting thresholds.

[7] Alternatively, we could fix two thresholds per indicator (e.g., to 0 and 1) to freely estimate the baseline intercepts and residual variances (see Mehta et al., 2004; Wu & Estabrook, 2016), which is consistent with the LISREL approach (Millsap & Tein, 2004).

```
>                          matL, matL0, matC1, matC2,
>                          matE, matE0, matD1, matD2,
>                          matP, matP0, matH1, matH2,
>                          matA, matA0, matG1, matG2,
>                          matIa, matIb, matV1, matV2,
>                          matVar, matR, matCov, matM,
>                          matC, matThres, expF,
>                          fitF, mxdata_ord)
> modUn_thres_con <- mxConstrainMLThresholds
>                 (modUn_thres)
```

We can then fit the model

```
> fitUn_thres <- mxRun(modUn_thres_con)
```

Note that the threshold structure discussed above makes the model more complex as compared with a MNLFA model for continuous indicators. Therefore, computation time for ordinal MNLFA models increases substantially, especially in the case of many indicators. In addition, optimization of the likelihood function is numerically more challenging which may cause the estimation to fail. For ordinal indicators, it is therefore advisable to use different starting values using the `mxTryHardOrdinal()` function. As compared with Mplus, `OpenMx` is slower in the case of ordinal indicators. Fitting the final model above to the indicators of the SI factor only takes up to 70 min in `OpenMx` while it takes about a minute in Mplus (on an Intel Core i5 with 8GB of RAM-memory). Advantage of `OpenMx` is however, that it is more flexible for models with ordinal indicators. For instance, residual variances can be moderated in `OpenMx` which is not possible in Mplus.

The analysis above assumes rather than tests invariance of thresholds. We could test that assumption[8] by comparing its fit with a MNLFA in which thresholds (but *not* intercepts or residual variances) are functions of background variables. But when $k = 3$, these models would be statistically equivalent because (effects on) two thresholds are interchangeable with (effects on) the intercept and residual variance, so threshold invariance can only be assumed and not tested (Wu & Estabrook, 2016). In the special case of binary indicators, fixing effects on only one threshold cannot identify effects on both the intercept and residual variance. Thus, threshold invariance still cannot be independently tested, but neither can invariance of factor loadings and intercepts (Wu & Estabrook, 2016). So for binary indicators, one could compare the fit of a configural model (with background-variable effects on factor loadings and intercepts, but not thresholds or residual variances) to a scalar-invariance model (with background-variable effects on residual variances, but not on thresholds, factor loadings, or intercepts). If $H_0$ can be rejected, partial invariance can still be established, but the data could not distinguish the nature of an indicator's DIF.

## Discussion

This article illustrated how to perform MNLFA for measurement invariance assessment using the R package `OpenMx`. We considered a two-factor MNLFA model for the DS14 data (Denollet et al., 2013) and showed how to evaluate full and partial measurement invariance with respect to a dichotomous and continuous background variable simultaneously. This is one of the first articles showing how to test for measurement invariance with MNLFA in free and open-source SEM software, cross-validating its results with the Mplus package. We therefore hope that with this tutorial we provide more researchers the opportunity to perform MNLFA for assessing measurement invariance or other purposes.

There are multiple advantages of MNLFA over other methods for testing measurement invariance. Unlike the MGCFA approach, MNLFA allows for the assessment of measurement invariance with respect to multiple background variables simultaneously. In addition, whereas MGCFA is only appropriate for categorical background variables, MNLFA also permits the assessment of measurement invariance with respect to continuous background variables. Whereas other single-group approaches (e.g., RFA and MIMIC) share these advantages, MNLFA does not require assuming common-factor or residual homoskedasticity with respect to the background variables. Although the use of product indicators to model moderation of factor loadings in RFA or MIMIC can capture some heteroskedasticity present in the data (Kolbe et al., 2021), it is not as flexible or interpretable as with MNLFA. We aimed to highlight all of these advantages in this tutorial by fitting an MNLFA model to empirical data that included all of these elements.

One of the steps included in this tutorial is selecting a set of anchor indicators. This step is useful for almost all methods for evaluating measurement invariance. A traditional strategy is to use all indicators other than the studied indicator as anchors, which leads to a contaminated set of anchors when one or more indicators in the anchor set violate measurement invariance. This can in turn lead to biased parameter estimates and inflated type I error rates when assessing measurement invariance (W.-C. Wang, 2004). It is therefore advisable to use an anchor-selection strategy to select a smaller subset of anchor indicators (preferably 10%–20% of the total number of indicators), such as the rank-based strategy proposed by Woods (2009). Several simulation studies have shown that this strategy frequently obtains uncontaminated sets of anchor indicators (Kolbe & Jorgensen, 2019; Kopf et al., 2015b; M. Wang & Woods, 2017; Woods, 2009). In this tutorial, we provided a step-by-step explanation of how to select anchor indicators using the rank-based strategy. If the user wants to prevent making a decision about which indicators should serve as anchors, combining MNLFA with a regularization approach may be an appealing method to assess measurement invariance (see Bauer et al., 2020).

The key element of the MNLFA approach is that CFA parameters are predicted by the background variables. As such, a functional form has to be assumed between the parameters and background variables. The present tutorial illustrated how to model (log-)linear relationships between the background variables and model parameters in MNLFA, which might not always be an accurate representation of the data. In such situations, a researcher could consider higher-order polynomial functions (Bauer & Hussong, 2009) or semiparametric MNLFA approaches including the local SEM approach by Hildebrandt et al. (2016), the mixture approach by Molenaar (2021), or the score based approach by Merkle and Zeileis (2013). The advantage of these semiparametric MNLFA approaches is that an assumption about the functional form between

[8] Note that although it is popular to test invariance of ordinal indicators by leaving intercepts fixed to zero and testing threshold invariance in place of intercept invariance (i.e., after testing invariance of loadings; Liu et al., 2017; Millsap & Tein, 2004), Wu and Estabrook (2016) explained why this leads to invalid comparisons (i.e., the response scales are not linked unless $\geq 2$ thresholds are equivalent across background variables).

the background variables and parameters is not required, making it a suitable approach for exploratory situations in which there is no theory about the functional form of the relationship.

One of the limitations of the current article is that we only demonstrated one method for assessing measurement invariance. Although this method seems to perform well across various conditions, many other methods are available as well. Therefore, we close by briefly mentioning the latest developments regarding methods for measurement invariance assessment and (open-source) software packages. From the semiparametric approaches discussed above, the local SEM approach can be applied using open-source R package `sirt` (Robitzsch, 2020a) and the score-based approach can be applied in open-source R package `lavaan` (Rosseel, 2012). The mixture approach is currently not implemented in an open-source package yet, but it can be applied using `OpenMx` in principle and it can readily be applied in Mplus (Muthén & Muthén, 2012). Examples of related measurement-invariance tools available in open-source software packages include the automated multigroup tests from the R package `semTools` (Jorgensen et al., 2019) and the multigroup DIF tests for categorical data from R packages `difR` (Magis et al., 2010) and `mirt` (Chalmers, 2012). In addition, the cluster-based approach to identify anchor items to test for measurement invariance with respect to a continuous variable (Schulze & Pohl, 2021) can be applied within R using the MNLFA implementation from the present article.

## References


Bauer, D. J. (2017). A more general model for testing measurement invariance and differential item functioning. *Psychological Methods*, *22*(3), 507–526. https://doi.org/10.1037/met0000077

Bauer, D. J., & Hussong, A. M. (2009). Psychometric approaches for developing commensurate measures across independent studies: Traditional and new models. *Psychological Methods*, *14*(2), 101–125. https://doi.org/10.1037/a0015583

Bauer, D. J., Belzak, W. C., & Cole, V. T. (2020). Simplifying the assessment of measurement invariance over multiple background variables: Using regularized moderated nonlinear factor analysis to detect differential item functioning. *Structural Equation Modeling*, *27*(1), 43–55. https://doi.org/10.1080/10705511.2019.1642754

Belzak, W. C. M. (2021). Regularized differential item functioning (Version 1.0.0) [Computer software manual]. https://github.com/wbelzak/regDIF/

Boker, S., Neale, M., Maes, H., Wilde, M., Spiegel, M., Brick, T., Spies, J., Estabrook, R., Kenny, S., Bates, T., Mehta, P., & Fox, J. (2011). OpenMx: An open source extended structural equation modeling framework. *Psychometrika*, *76*(2), 306–317. https://doi.org/10.1007/S11336-010-9200-6

Chalmers, R. P. (2012). A multidimensional item response theory package for the R environment. *Journal of Statistical Software*, *48*(6), 1–29. https://doi.org/10.18637/jss.v048.i06

Denollet, J. (2005). DS14: Standard assessment of negative affectivity, social inhibition, and Type D personality. *Psychosomatic Medicine*, *67*(1), 89–97. https://doi.org/10.1097/01.psy.0000149256.81953.49

Denollet, J., Pedersen, S. S., Vrints, C. J., & Conraads, V. M. (2013). Predictive value of social inhibition and negative affectivity for cardiovascular events and mortality in patients with coronary artery disease: the type D personality construct. *Psychosomatic Medicine*, *75*(9), 873–881. https://doi.org/10.1097/PSY.0000000000000001

Hallquist, M. N., & Wiley, J. F. (2018). MplusAutomation: An R package for facilitating large-scale latent variable analyses in Mplus. *Structural Equation Modeling*, *25*(4), 621–638. https://doi.org/10.1080/10705511.2017.1402334

Hildebrandt, A., Lüdtke, O., Robitzsch, A., Sommer, C., & Wilhelm, O. (2016). Exploring factor model parameters across continuous variables with local structural equation models. *Multivariate Behavioral Research*, *51*(2–3), 257–258. https://doi.org/10.1080/00273171.2016.1142856

Horn, J. L., & McArdle, J. J. (1992). A practical and theoretical guide to measurement invariance in aging research. *Experimental Aging Research*, *18*(3–4), 117–144. https://doi.org/10.1080/03610739208253916

Jöreskog, K. G., & Goldberger, A. S. (1975). Estimation of a model with multiple indicators and multiple causes of a single latent variable. *Journal of the American Statistical Association*, *70*(351a), 631–639. https://doi.org/10.1080/01621459.1975.10482485

Jorgensen, T. D. (2017). Applying permutation tests and multivariate modification indices to configurally invariant models that need respecification. *Frontiers in Psychology*, *8*, 1455. https://doi.org/10.3389/fpsyg.2017.01455

Jorgensen, T. D., Pornprasertmanit, S., Schoemann, A. M., & Rosseel, Y. (2019). semTools: Useful tools for structural equation modeling (R package version 0.5-1.935) [Computer software manual]. https://CRAN.R-project.org/package=semTools

Kolbe, L., & Jorgensen, T. D. (2019). Using restricted factor analysis to select anchor items and detect differential item functioning. *Behavior Research Methods*, *51*(1), 138–151. https://doi.org/10.3758/s13428-018-1151-3

Kolbe, L., Jorgensen, T. D., & Molenaar, D. (2021). The impact of unmodeled heteroskedasticity on assessing measurement invariance in single-group models. *Structural Equation Modeling*, *28*(1), 82–98. https://doi.org/10.1080/10705511.2020.1766357

Kopf, J., Zeileis, A., & Strobl, C. (2015a). Anchor selection strategies for DIF analysis: Review, assessment, and new approaches. *Educational and Psychological Measurement*, *75*(1), 22–56. https://doi.org/10.1177/0013164414529792

Kopf, J., Zeileis, A., & Strobl, C. (2015b). A framework for anchor methods and an iterative forward approach for DIF detection. *Applied Psychological Measurement*, *39*(2), 83–103. https://doi.org/10.1177/0146621614544195

Liu, Y., Millsap, R. E., West, S. G., Tein, J.-Y., Tanaka, R., & Grimm, K. J. (2017). Testing measurement invariance in longitudinal data with ordered-categorical measures. *Psychological Methods*, *22*(3), 486–506.

Magis, D., Béland, S., Tuerlinckx, F., & De Boeck, P. (2010). A general framework and an r package for the detection of dichotomous differential item functioning. *Behavior Research Methods*, *42*(3), 847–862. https://doi.org/10.3758/BRM.42.3.847

Mehta, P. D., Neale, M. C., & Flay, B. R. (2004). Squeezing interval change from ordinal panel data: latent growth curves with ordinal outcomes. *Psychological Methods*, *9*(3), 301–333.

Mellenbergh, G. J. (1989). Item bias and item response theory. *International Journal of Educational Research*, *13*(2), 127–143. https://doi.org/10.1016/0883-0355(89)90002-5

Meredith, W. (1993). Measurement invariance, factor analysis and factorial invariance. *Psychometrika*, *58*(4), 525–543. https://doi.org/10.1007/BF02294825

Merkle, E. C., & Zeileis, A. (2013). Tests of measurement invariance without subgroups: A generalization of classical methods. *Psychometrika*, *78*(1), 59–82. https://doi.org/10.1007/S11336-012-9302-4

Millsap, R. E., & Tein, J.-Y. (2004). Assessing factorial invariance in ordered-categorical measures. *Multivariate Behavioral Research*, *39*(3), 479–515. https://doi.org/10.1207/S15327906MBR3903_4

Molenaar, D. (2021). A flexible moderated factor analysis approach to test for measurement invariance across a continuous variable. *Psychological Methods*, *26*(6), 660–679. https://doi.org/10.1037/met0000360

Muthén, B. (1984). A general structural equation model with dichotomous, ordered categorical, and continuous latent variable indicators. *Psychometrika*, *49*(1), 115–132. https://doi.org/10.1007/BF02294210

Muthén, B., & Asparouhov, T. (2002). Latent variable analysis with categorical outcomes: Multiple-group and growth modeling in Mplus. *Mplus Web Notes*, *4*(5), 1–22.

Muthén, L. K., & Muthén, B. O. (2012). *Mplus user's guide* (7th ed.).

Neale, M. C., Hunter, M. D., Pritikin, J. N., Zahery, M., Brick, T. R., Kirkpatrick, R. M., Estabrook, R., Bates, T. C., Maes, H. H., & Boker, S. M. (2016). OpenMx 2.0: Extended structural equation and statistical modeling. *Psychometrika*, *81*(2), 535–549. https://doi.org/10.1007/s11336-014-9435-8

Oort, F. J. (1992). Using restricted factor analysis to detect item bias. *Methodika*, *6*, 150–166.

Putnick, D. L., & Bornstein, M. H. (2016). Measurement invariance conventions and reporting: The state of the art and future directions for psychological research. *Developmental Review*, *41*, 71–90.

R Core Team. (2018). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing. https://www.R-project.org/

Rhemtulla, M., Brosseau-Liard, P. É., & Savalei, V. (2012). When can categorical variables be treated as continuous? A comparison of robust continuous and categorical SEM estimation methods under suboptimal conditions. *Psychological Methods*, *17*(3), 354–373. https://doi.org/10.1037/a0029315

Robitzsch, A. (2019). Moderated nonlinear factor analysis (R package version 0.1-53) [Computer software manual]. https://CRAN.R-project.org/package=mnlfa

Robitzsch, A. (2020a). Supplementary item response theory models (R package version 3.9-4) [Computer software manual]. https://CRAN.R-project.org/package=sirt

Robitzsch, A. (2020b). Why ordinal variables can (almost) always be treated as continuous variables: Clarifying assumptions of robust continuous and ordinal factor analysis estimation methods. *Frontiers in Education*, *5*, 177. https://doi.org/10.3389/feduc.2020.589965

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

Schauberger, G. (2021). GPCMlasso: Differential item functioning in generalized partial credit models (R package version 0.1-5) [Computer software manual]. https://CRAN.R-project.org/package=GPCMlasso

Schulze, D., & Pohl, S. (2021). Finding clusters of measurement invariant items for continuous covariates. *Structural Equation Modeling: A Multidisciplinary Journal*, *28*(2), 219–228. https://doi.org/10.1080/10705511.2020.1771186

Stan Development Team. (2021). RStan: The R interface to Stan (R package version 2.21.3) [Computer software]. https://mc-stan.org/

Stark, S., Chernyshenko, O. S., & Drasgow, F. (2006). Detecting differential item functioning with confirmatory factor analysis and item response theory: Toward a unified strategy. *Journal of Applied Psychology*, *91*(6), 1292–1306. https://doi.org/10.1037/0021-9010.91.6.1292

Steenkamp, J.-B. E., & Baumgartner, H. (1998). Assessing measurement invariance in cross-national consumer research. *Journal of Consumer Research*, *25*(1), 78–90. https://doi.org/10.1086/209528

Takane, Y., & De Leeuw, J. (1987). On the relationship between item response theory and factor analysis of discretized variables. *Psychometrika*, *52*(3), 393–408. https://doi.org/10.1007/BF02294363

Van der Ark, L. A. (2007). Mokken scale analysis in R. *Journal of Statistical Software*, *20*(11), 1–19. https://doi.org/10.18637/jss.v020.i11

Vandenberg, R. J., & Lance, C. E. (2000). A review and synthesis of the measurement invariance literature: Suggestions, practices, and recommendations for organizational research. *Organizational Research Methods*, *3*(1), 4–70. https://doi.org/10.1177/109442810031002

Wang, M., & Woods, C. M. (2017). Anchor selection using the Wald test anchor-all-test-all procedure. *Applied Psychological Measurement*, *41*(1), 17–29. https://doi.org/10.1177/0146621616668014

Wang, W.-C. (2004). Effects of anchor item methods on the detection of differential item functioning within the family of Rasch models. *The Journal of Experimental Education*, *72*(3), 221–261. https://doi.org/10.3200/JEXE.72.3.221-261

Wirth, R., & Edwards, M. C. (2007). Item factor analysis: current approaches and future directions. *Psychological Methods*, *12*(1), 58–79.

Woods, C. M. (2009). Empirical selection of anchors for tests of differential item functioning. *Applied Psychological Measurement*, *33*(1), 42–57. https://doi.org/10.1177/0146621607314044

Wu, H., & Estabrook, R. (2016). Identification of confirmatory factor analysis models of different levels of invariance for ordered categorical outcomes. *Psychometrika*, *81*(4), 1014–1045. https://doi.org/10.1007/s11336-016-9506-0

Received November 12, 2021
Revision received February 28, 2022
Accepted March 1, 2022 ■