xxm Reference Manual

February 21, 2013

Type Package

Title Structural Equation Modeling for Dependent Data

Version 0.5.0

Date 2013-02-06

Author Paras Mehta

Depends

Maintainer <paras.mehta@times.uh.edu>

Description Structural Equation Modeling with complex dependent data structures

License XXM

LazyLoad yes

R topics documented:

brim.school .......................................................... 3
brim.student .......................................................... 3
ex98.11 .................................................................. 4
ex98.12 .................................................................. 5
faces.rater ............................................................... 5
faces.response .......................................................... 6
faces.target .............................................................. 7
faces.xxm.1 .............................................................. 7
gcsemv.boy .............................................................. 11
gcsemv.girl ............................................................... 12
gcsemv.school .......................................................... 12
gcsemv.xxm.1 ............................................................ 13
gcsemv.xxm.2 ............................................................ 17
hcfa.school ............................................................... 20
hcfa.student ............................................................. 20
hcfa.teacher ............................................................. 21
**brim.school**

**XXM Dataset: brim.school**

**Description**

brim contains two data frames: brim.student and brim.school, that have been formatted for xxM analysis. The data were simulated to represent a common research design in educational research.

**Usage**

data(brim.school)

**Format**

A data frame with 25 observations on the following variable.

- school a numeric vector

**See Also**

brim.student

**Examples**

data(brim.school)

## maybe str(brim.school); plot(brim.school) ...

---

**brim.student**

**XXM Dataset: brim.student**

**Description**

brim contains two data frames: brim.student and brim.school, that have been formatted for xxM analysis. The data were simulated to represent a common research design in educational research.

**Usage**

data(brim.student)

**Format**

A data frame with 75 observations on the following 4 variables.

- student a numeric vector
- school a numeric vector
- y1 a numeric vector
- y2 a numeric vector
See Also

`brim.school`

Examples

data(brim.student)
## maybe str(brim.student) ; plot(brim.student) ...

---

**ex98.11**

*XXM Dataset: ex98.11*

---

**Description**

*ex98* consists of two simulated data sets generated from a population model mirroring Example 9.8 in the MPlus Version 6 User’s Guide (Muthen & Muthen, 2011): *ex98.11* and *ex98.12*.

**Usage**

data(ex98.11)

**Format**

A data frame with 300 observations on the following 8 variables.

- `l1` a numeric vector
- `l2` a numeric vector
- `y1` a numeric vector
- `y2` a numeric vector
- `y3` a numeric vector
- `y4` a numeric vector
- `x1` a numeric vector
- `x2` a numeric vector

See Also

*ex98.12*

Examples

data(ex98.11)
## maybe str(ex98.11) ; plot(ex98.11) ...
**ex98.12**

**XXM Dataset: ex98.12**

**Description**

ex98 consists of two simulated data sets generated from a population model mirroring Example 9.8 in the MPlus Version 6 User’s Guide (Muthen & Muthen, 2011): **ex98.11** and **ex98.12**.

**Usage**

```r
data(ex98.12)
```

**Format**

A data frame with 50 observations on the following 2 variables.

- **a** a numeric vector
- **w** a numeric vector

**See Also**

**ex98.11**

**Examples**

```r
data(ex98.12)
## maybe str(ex98.12) ; plot(ex98.12) ...
```

---

**faces.rater**

**XXM Dataset: faces.rater**

**Description**

faces contains three data frames: **faces.response**, **faces.rater** and **faces.target**, that have been formatted for xxM analysis. These data were drawn from a study in which participants provided symmetry and attractiveness ratings for 15 randomly selected facial photographs.

**Usage**

```r
data(faces.rater)
```

**Format**

A data frame with 243 observations on the following variable.

- **rater** a numeric vector
See Also

faces.response; faces.target

Examples

data(faces.rater)
## maybe str(faces.rater) ; plot(faces.rater) ...

faces.response

XXM Dataset: faces.response

Description

faces contains three data frames: faces.response, faces.rater and faces.target, that have been formatted for xxM analysis. These data were drawn from a study in which participants provided symmetry and attractiveness ratings for 15 randomly selected facial photographs.

Usage

data(faces.response)

Format

A data frame with 3600 observations on the following 5 variables.

response  a numeric vector
rater     a numeric vector
target    a numeric vector
SYM       a numeric vector
PA        a numeric vector

See Also

faces.rater; faces.target

Examples

data(faces.response)
## maybe str(faces.response) ; plot(faces.response) ...
Description

faces contains three data frames: faces.response, faces.rater and faces.target, that have been formatted for xxM analysis. These data were drawn from a study in which participants provided symmetry and attractiveness ratings for 15 randomly selected facial photographs.

Usage

    data(faces.target)

Format

A data frame with 39 observations on the following variable.

    target  a numeric vector

See Also

    faces.rater; faces.response

Examples

    data(faces.target)
    ## maybe str(faces.target) ; plot(faces.target) ...

Introduction

faces is a three level SEM model for rater-target research designs: participants (raters) respond to a series of randomly sampled stimuli (targets). Specifically, responses are simultaneously nested within raters and targets, leading to a cross-classified dependency structure. Treating these sources of non-independence as random effects allows the results to generalize to the larger populations from which these stimuli were drawn (Judd, Westfall, & Kenny, 2012). In the present example, we use data collected from a sample of 243 undergraduate students who evaluated the symmetry (SYM) and physical attractiveness (PA) of photographs featuring male and female faces (Langner et al., 2010). The first model features a decomposition of SYM and PA ratings into rater- and target-specific variance components, and latent correlations between PA and SYM for each of these sources. Response-specific residuals will also be allowed to correlate. Unlike the standard analytic approach which confounds these distinct sources of variability in photo ratings, the current approach answers precise research questions that are specific to each class of variance component. Specifically, the correlation between rater variance components describes the extent to which individuals
who evaluate all photographs as consistently more or less symmetrical, tend to also rate all photographs as more or less attractive. Moreover, the correlation between target variance components expresses the degree to which target photographs who are evaluated by all raters as consistently more or less symmetrical, are also rated as more or less attractive. Finally, the correlation between response-specific residuals reflects idiosyncratic associations between symmetry and attractiveness.

**Creating a model: ‘faces’**

The first model contains three levels: **response**, **rater**, **target**

1. **response** level is nested within **rater** and **target** and includes two endogenous (dependent) variables: **SYM** (symmetry) and **PA** (attractiveness).

2. **rater** is a ‘parent’ of **response** and has no observed dependent or independent variables, but includes a two latent variables: **rater_SYM** and **rater_PA**. These latent variables measure the extent to which participants (rater) evaluate all targets as consistently more or less symmetrical/attractive. For example, raters who score ‘high’ on **rater_SYM** exhibit a tendency to see all targets as more symmetrical than average. These rater effects could also stem from individual differences in implicit preferences for one end of the response scale.

3. **target** is a ‘parent’ of **response** and has no observed dependent or independent variables, but includes a two latent variables: **target_SYM** and **target_PA**. These latent variables measure the extent to which target faces (photographs) are evaluated as consistently more or less symmetrical/attractive by all raters. For example, a target that score ‘high’ on **target_SYM** are rated as more symmetrical than average by all raters.

**faces** is created by invoking `xxmModel()`.

```r
faces <- xxmModel(levels = c("response","rater","target"))
```

**Adding submodels: ‘response’, ‘rater’ and ‘target’**

For each level declared above, we need to create corresponding submodels and link these to the level-specific data frames. A submodel is created by invoking `xxmSubmodel()`:

1. **response**

   ```r
   faces <- xxmSubmodel(model = faces, level = "response",
                       parents = c("rater","target"), ys = c("SYM","PA"),
                       xs = , etas = , data = faces.response)
   ```

2. **rater**

   ```r
   faces <- xxmSubmodel(model = faces, level = "rater",
                       parents = , ys = , xs = , etas = c("rater_SYM","rater_PA"),
                       data = faces.rater)
   ```

3. **target**

   ```r
   faces <- xxmSubmodel(model = faces, level = "target",
                       parents = , ys = , xs = , etas = c("target_SYM","target_PA"),
                       data = faces.target)
   ```
Specifying model matrices for response

With a two observed dependent variables SYM and PA, the submodel for response rather straightforward. It includes the residual variance (theta) and intercept (nu) of the dependent variables. It is necessary to create pattern, value, and label matrices for each model matrix. For example the theta and nu matrices at the response level require the following R matrices to be created:

```
resp_th_pat <- matrix(c(1,1,1,1),2,2)
resp_th_val <- matrix(c(2,./zero.noslash,./zero.noslash,2),2,2)
resp_th_lab <- matrix(c("SYM_Var","SYM_PA_Cov","SYM_PA_Cov","PA_Var"),2,2)
resp_nu_pat <- matrix(c(1,1),2,1)
resp_nu_val <- matrix(c(5,5),2,1)
resp_nu_lab <- matrix(c("SYM_Int","PA_Int"),2,1)
```

These R matrices are combined to create model matrices for each submodel using xxmWithinMatrix() command:

1. **theta**: Observed residual-covariance matrix is a (2x2) matrix with a three non-redundant elements: residual variances for SYM and PA, along with the covariance.

```
faces <- xxmWithinMatrix(model = faces, level = "response", type = "theta",
                    pattern = resp_th_pat, value = resp_th_val, label = resp_th_lab)
```

2. **nu**: Observed variable Intercept matrix is a (2x1) matrix with a two elements: the intercepts for SYM and PA.

```
faces <- xxmWithinMatrix(model = faces, level = "response",
                    type = "nu", pattern = resp_nu_pat, value = resp_nu_val, label = resp_nu_lab)
```

Specifying model matrices for rater

The submodel for rater contains two latent variables: rater_SYM and rater_PA, and no observed variables. Because mean-structure has been modeled at the response level, the it is only necessary to specify a variance-covariance matrix for this level.

The latent variable covariance (psi) matrix for the rater level is comprised of two rows and two columns, one for each latent variable:

```
rater_psi_pat <- matrix(c(1,1,1,1),2,2) rater_psi_val <- matrix(c(1,.01,.01,1),2,2)
rater_psi_lab <- matrix(c("SYM_Rater_Var","SYM_PA_Rater_Cov",
                         "SYM_PA_Rater_Cov","PA_Rater_Var"),2,2)
```

1. **psi**: Latent variable covariance matrix

```
faces <- xxmWithinMatrix(model = faces, level = "rater", type = "psi",
                    pattern = rater_psi_pat, value = rater_psi_val, label = rater_psi_lab)
```

Specifying model matrices for target

The submodel for target has two latent variables: target_SYM and target_PA and no observed variables.

As before, the latent variable covariance (psi) matrix is comprised of two rows and two columns, one for each latent variable:

```
target_psi_pat <- matrix(c(1,1,1,1),2,2) target_psi_val <- matrix(c(1,.01,.01,1),2,2)
```

1. **psi**: Latent variable covariance matrix

```
faces <- xxmWithinMatrix(model = faces, level = "target", type = "psi",
                    pattern = target_psi_pat, value = target_psi_val, label = target_psi_lab)
```
target_psi_lab <- matrix(c("SYM_Target_Var", "SYM_PA_Target_Cov", "SYM_PA_Target_Cov", "PA_Target_Var"),2,2)

1. psi: Latent variable covariance matrix

faces <- xxmWithinMatrix(model = faces, level = "target", type = "psi", pattern = target_psi_pat, value = target_psi_val, label = target_psi_lab)

Connecting response to rater and target

Now that the within-level parameters have been specified for each submodel, we can begin specifying the relationships among observed and latent variables across levels. In the present model, the observed variables SYM and PA at the response level, serve as indicators has a latent factors at the rater and target levels. These measurement relationships are specified by regressing the observed outcome variables at the response level on the appropriate latent variables at the rater and target levels, and fixing the coefficients to 1.0 (Mehta & Neale, 2005). As before, we will use label, pattern, and value matrices to create the appropriate model matrices using the xxmBetweenMatrix() command.

Specifying across-level model connecting rater (level-2) with response (level-1)

The between-levels factor-loading (lambda) matrix has 2 rows and 2 columns: one column for each latent variable, and one for each observed variable. All of the elements of this matrix are fixed, and each response level indicator loads on it’s respective rater level latent variable:

rater_resp_la_pat <- matrix(c(0,0,0,0),2,2) rater_resp_la_val <- matrix(c(1,0,0,1),2,2) rater_resp_la_lab <- matrix(c("SYM_Rater_Loading","NA","NA","PA_Rater_Loading"),2,2)

1. psi: Latent variable covariance matrix

faces <- xxmBetweenMatrix(model = faces, parent = "rater", child = "response", type = "lambda", pattern = rater_resp_la_pat, value = rater_resp_la_val, label = rater_resp_la_lab)

Specifying across-level model connecting target (level-3) with response (level-1)

The between-levels factor-loading (lambda) matrix has 2 rows and 2 columns: one column for each latent variable, and one for each observed variable. All of the elements of this matrix are fixed, and each response level indicator loads on it’s respective target level latent variable:

target_resp_la_pat <- matrix(c(0,0,0,0),2,2) target_resp_la_val <- matrix(c(1,0,0,1),2,2) target_resp_la_lab <- matrix(c("SYM_Target_Loading","NA","NA","PA_Target_Loading"),2,2)

1. psi: Latent variable covariance matrix

faces <- xxmBetweenMatrix(model = faces, parent = "target", child = "response", type = "lambda", pattern = target_resp_la_pat, value = target_resp_la_val, label = target_resp_la_lab)
Compute: xxmRun()

Model specification is now complete. Parameter estimation is initiated:

```r
faces <- xxmRun(model = faces)
```

Note

The code for this model may be found in and the data frames can be loaded by invoking: `data(faces.xxm)`

---

gcsemv.boy  

**XXM Dataset: gcsemv.boy**

---

**Description**

gcsemv is comprised of 3 data frames: `gcsemv.boy` and `gcsemv.girl` contain writing and coursework scores for a sample of male and female students, and `gcsemv.school` contains ids for schools.

**Usage**

```r
data(gcsemv.boy)
```

**Format**

A data frame with 624 observations on the following 4 variables.

- `boy` a numeric vector
- `school` a numeric vector
- `written` a numeric vector
- `coursework` a numeric vector

**See Also**

`gcsemv.girl`; `gcsemv.school`

**Examples**

```r
data(gcsemv.boy)
## maybe str(gcsemv.boy) ; plot(gcsemv.boy) ...```
**gcsemv.girl**  
*XXM Dataset: gcsemv.girl*

**Description**

gcsemv is comprised of 3 data frames: **gcsemv.boy** and **gcsemv.girl** contain writing and coursework scores for a sample of male and female students, and **gcsemv.school** contains ids for schools.

**Usage**

```r
data(gcsemv.girl)
```

**Format**

A data frame with 899 observations on the following 4 variables.

- **girl** a numeric vector
- **school** a numeric vector
- **written** a numeric vector
- **coursework** a numeric vector

**See Also**

**gcsemv.boy**; **gcsemv.school**

**Examples**

```r
data(gcsemv.girl)  
## maybe str(gcsemv.girl) ; plot(gcsemv.girl) ...
```

**gcsemv.school**  
*XXM Dataset: gcsemv.school*

**Description**

gcsemv is comprised of 3 data frames: **gcsemv.boy** and **gcsemv.girl** contain writing and coursework scores for a sample of male and female students, and **gcsemv.school** contains ids for schools.

**Usage**

```r
data(gcsemv.school)
```

**Format**

A data frame with 73 observations on the following variable.

- **school** a numeric vector
See Also

gcsemv.girl; gcsemv.boy

Examples

data(gcsemv.school)
## maybe str(gcsemv.school) ; plot(gcsemv.school) ...

Introduction

gcsemv is a two level SEM model for boys and girls within schools. Each student has two scores, one on a written test and the other on coursework. The current model is the most unrestricted model possible which allows for several associations. First, schools differ in the degree to which (1) their boys perform on the written test, (2) their boys perform on coursework, (3) their girls perform on the written test, and (4) their girls perform on coursework. These effects of school are not restricted in any way, so it is possible for a school to only teach boys well in writing and poorly in the other three categories or to have boys and girls with high coursework scores and low writing scores, etc. However, the associations between these are estimated as they very likely correlate (better schools will likely be better at teaching all students). Furthermore, this model allows boys and girls to be different. Specifically, they can have different means for the variables showing that boys and girls can have different predispositions towards succeeding on one test versus another. Second, variability in each outcome and the correlation between the two may be different across boys and girls.

This example also illustrates an interesting and important feature of xxM. Gender is a fixed attribute of person. Hence, boys and girls would be treated as students belonging to two different groups defined by gender. However, when separate covariance structure is specified across two groups for all practical purpose the two may be thought of as different populations. The fact that both are students is besides the point. In xxM observations from the ‘same level’ that belong to groups defined by a fixed attribute are treated as different levels.

Creating a model: ‘gcsemv’

The model involves three levels: boy, girl, school

1. boy is nested within school. boy is said to be the child level with school as the corresponding parent level.
   boy has two observed dependent variable: written and course.
2. girl is nested within school. girl is said to be the child level with school as the corresponding parent level.
   girl has two observed dependent variable: written and course.
3. school is not nested within any level and is a parent of both boy and girl.
   school has four latent variables:
(a) **boywritten** is the school level random intercept of response **written** at the **boy** level.
(b) **boycourse** is the school level random intercept of response **course** at the **boy** level.
(c) **girlwritten** is the school level random intercept of response **written** at the **girl** level.
(d) **girlcourse** is the school level random intercept of response **course** at the **girl** level.

The model is created by invoking `xxmModel()`:

```r
gcsemv <- xxmModel(levels = c("boy","girl","school"))
```

Note: Levels are numbered according to their position in the above command. Hence, **boy** corresponds to level 1, **girl** corresponds to level 2, and **school** corresponds to level 3.

It may seem odd that the levels **girl** and **boy** are numbered 1 and 2, respectively. These levels are at equivalent levels, or are **siblings**. Because this, the order of these levels could be entered in the other way in which **girl** is level 1 and **boy** is level 2 without changing the model. Later statements will clarify the structure of the datasets so both these models will be equal. However, it is essential that **school** is at level 3.

Adding submodels for ‘boy’, ‘girl’ and ‘school’

For each level declared above, we need to create corresponding submodels. A submodel is created by invoking `xxmSubmodel()`:

1. **boy**
   ```r
gcsemv <- xxmSubmodel(model = gcsemv, level = "boy", parents = "school",
   ys = c("written","course"), xs = , etas = , data = gcsemv.boy)
   ```

2. **girl**
   ```r
gcsemv <- xxmSubmodel(model = gcsemv, level = "girl", parents = "school",
   ys = c("written","course"), xs = , etas = , data = gcsemv.girl)
   ```

3. **school**
   ```r
gcsemv <- xxmSubmodel(model = gcsemv, level = "school",
   parents = , ys = , xs = , etas =c("boywritten","boycourse",
   "girlwritten","girlcourse"), data = gcsemv.school)
   ```

Specifying within-level model matrices for ‘boy’ (level 1)

The boy submodel is fairly simple with two observed variables (**written** and **course**). Model matrices for each submodel is specified by invoking `xxmWithinMatrix()`:

1. **theta**: Observed residual-covariance matrix is a (2x2) matrix with a three unique elements, the residual variance of **written**, the residual variance of **course**, and the residual covariance between **written** and **course**.
   ```r
gcsemv <- xxmWithinMatrix(model = gcsemv, level = "boy",
   type = "theta", pattern = th1_pat, value = th1_val)
   ```

2. **nu**: Observed variable intercept matrix is a (2x1) matrix with two elements, the intercept of **written** and the intercept of **course**.
   ```r
gcsemv <- xxmWithinMatrix(model = gcsemv, level = "boy",
   type = "nu", pattern = nul_pat, value = nul_val)
   ```
Specifying within-level model matrices for ‘girl’ (level 2)

The girl submodel the mirror image of the boy submodel. However, these parameters will be estimated separately. girl also has two observed variables (written and course).

1. **theta**: Observed residual-covariance matrix is a \((2 \times 2)\) matrix with a three unique elements, the residual variance of written, the residual variance of course, and the residual covariance between written and course.

   \[
   \text{gcsemv} <- \text{xxmWithinMatrix(model = gcsemv, level = "girl", type = "theta", pattern = th2\_pat, value = th2\_val)}
   \]

2. **nu**: Observed variable intercept matrix is a \((2 \times 1)\) matrix with two elements, the intercept of written and the intercept of course.

   \[
   \text{gcsemv} <- \text{xxmWithinMatrix(model = gcsemv, level = "girl", type = "nu", pattern = nu2\_pat, value = nu2\_val)}
   \]

Specifying within-level model matrices for ‘school’ (level 3)

school has four latent variables boywritten, boycourse, girlwritten, and girlcourse. This level includes a single latent covariance matrix.

1. **psi**: Latent covariance matrix is a \((4 \times 4)\) matrix. The elements include variances of the four latent variables and the six unique covariances between them.

   \[
   \text{gcsemv} <- \text{xxmWithinMatrix(model = gcsemv, level = "school", type = "psi", pattern = ps3\_pat, value = ps3\_val)}
   \]

Connecting ‘school’ to ‘boy’ and ‘girl’

So far, we have specified within-level matrices. We now need to connect observed and latent variables across-levels. Observed variables written and course are measured at both of the lowest levels: boy and girl. Each of these four observed variables has its own latent random-intercept at the school level. In other words, a single observed dependent variable within boy or girl is regressed on a single latent independent variable within school with a fixed coefficient of 1.0. This across-level regression connecting variables across two submodels is specified by invoking xxmBetweenMatrix().

Note: The direction of influence is from the higher level model school to the lower level model boy and girl. Number of rows of a between matrix equals the number of dependent variables for the child level. Number of columns of a between matrix equals the number of independent variables for the parent level. In this case, the parent model has four latent variables and each of the child models has two observed variable. Hence, the lambda matrix connecting school with boy and girl submodel has four rows and two columns.

Specifying across-level model matrices connecting ‘school’ to ‘boy’

school has four latent variables and boy has two observed variables. The first two school latent variables boywritten and boycourse are measured by the corresponding observed variables for boys: written and course, respectively.
1. **lambda**: Factor-loading matrix is a (2x4) matrix. It has eight different elements, six of which are set to 0 and two of which are set to 1.0.

   gcsemv <- xxmBetweenMatrix(model = gcsemv, parent = "school", child = "boy", type = "lambda", pattern = ly13_pat, value = ly13_val)

   Note: The first row of the lambda matrix corresponds to the first variable in boy (i.e., written) and is loaded onto the first latent variable of school. We have called this latent variable boywritten, but until we specify this path between the levels, the variable has no meaning. Thus, it is essential to properly specify the between level matrices such that the label given to latent variables corresponds to the proper indicators.

**Specifying across-level model matrices connecting 'school' to 'girl'**

school has four latent variables and girl has two observed variables. The last two school latent variables girlwritten and girlcourse are measured by the corresponding observed variables for girls: written and course, respectively.

1. **lambda**: Factor-loading matrix is a (2x4) matrix. It has eight different elements, six of which are set to 0 and two of which are set to 1.0.

   gcsemv <- xxmBetweenMatrix(model = gcsemv, parent = "school", child = "girl", type = "lambda", pattern = ly23_pat, value = ly23_val)

**Compute: xxmRun()**

Model specification is now complete. Parameter estimation is initiated by:

   gcsemv <- xxmRun(model = gcsemv)

**Notes**

1. Dataset for this model is packaged with xmx in an R workspace called gcsemv.xxm.RData. The three data frames called ‘boy’, ‘girl’, and ‘school’ are loaded into the R workspace when the library is loaded:

   library(xxm)

   data(gcsemv.xxm, package="xxm")

2. Datasets for this model are documented under.

3. R script for running this model is stored under .../models/gcsemv.1.xxm.R

**See Also**

   gcsemv.xxm.2
gcsemv.xxm.2

Introduction

**gcsemv** is a two level SEM model for boys and girls within schools. Each student has two scores, one on a written test and the other on coursework. The current model is a restricted version of another model (gcsemv.xxm.1). In this model, the within model residual covariances and intercepts are constrained to be equal across **boy** and **girl**. However, covariances among random-intercepts at **school** for boys and girls are allowed to be different. Thus, there are still four school level variables: (1) random-intercept for boy’s written test, (2) random-intercept for boy’s coursework,(3) random-intercept for girl’s written test, (4) random-intercept for girls’s coursework.

Creating a model: ‘gcsemv’

The model involves three levels: **boy**, **girl**, **school**

1. **boy** is nested within **school**. **boy** is said to be the **child** level with **school** as the corresponding **parent** level. **boy** has two observed dependent variable: **written** and **course**.
2. **girl** is nested within **school**. **girl** is said to be the **child** level with **school** as the corresponding **parent** level. **girl** has two observed dependent variable: **written** and **course**.
3. **school** is not nested within any level and is a **parent** of both **boy** and **girl**. **school** has four latent variables:
   (a) **boywritten** is the school level random intercept of response **written** at the **boy** level.
   (b) **boycourse** is the school level random intercept of response **course** at the **boy** level.
   (c) **girlwritten** is the school level random intercept of response **written** at the **girl** level.
   (d) **girlcourse** is the school level random intercept of response **course** at the **girl** level.

The model is created by invoking **xxmModel()**.

`gcsemv <- xxmModel(levels = c("boy","girl","school"))`

Note: Levels are numbered according to their position in the above command. Hence, **boy** corresponds to level 1, **girl** corresponds to level 2, and **school** corresponds to level 3.

Adding submodels for ‘boy’, ‘girl’, and ‘school’

For each level declared above, we need to create corresponding submodels. Each submodel contains all the elements that will exist within a level. A submodel is created by invoking **xxmSubmodel()**: 

1. **boy**

   `gcsemv <- xxmSubmodel(model = gcsemv, level = "boy", parents = "school", ys = c("written","course"), xs = , etas = , data = boy)`

2. **girl**
3. **school**

gcsemv <- xxmSubmodel(model = gcsemv, level = "school", parents = , ys = , xs = , etas =c("boywritten", "boycourse","girlwritten","girlcourse"), data = school)

### Specifying within-level model matrices for ‘boy’ (level 1)

The boy submodel is fairly simple having two observed variables (**written** and **course**). Model matrices for each submodel is specified by invoking `xxmWithinMatrix()`:

1. **theta**: Observed residual-covariance matrix is a (2x2) matrix with a three unique elements, the residual variance of **written**, the residual variance of **course**, and the residual covariance between **written** and **course**.

   gcsemv <- xxmWithinMatrix(model = gcsemv, level = "boy", type = "theta", pattern = th1_pat, value = th1_val,label = th1_lab)

2. **nu**: Observed variable intercept matrix is a (2x1) matrix with two elements, the intercept of **written** and the intercept of **course**.

   gcsemv <- xxmWithinMatrix(model = gcsemv, level = "boy", type = "nu", pattern = nu1_pat, value = nu1_val, label = nu1_lab)

### Specifying within-level model matrices for ‘girl’ (level 2)

The girl submodel the mirror image of the boy submodel. Furthermore, we wish to constrain these two within matrices to be the same. To that end we use the same input matrices as we did within the **boy** level. The key to this specification is for the two models to share the same labels for each parameter. This is accomplished by using the same label matrix (th1_lab and nu1_lab) Model matrices for each submodel is specified by invoking `xxmWithinMatrix()`:

1. **theta**: Observed residual-covariance matrix is a (2x2) matrix with a three unique elements, the residual variance of **written**, the residual variance of **course**, and the residual covariance between **written** and **course**.

   gcsemv <- xxmWithinMatrix(model = gcsemv, level = "girl", type = "theta", pattern = th1_pat, value = th1_val,label = th1_lab)

2. **nu**: Observed variable intercept matrix is a (2x1) matrix with two elements, the intercept of **written** and the intercept of **course**.

   gcsemv <- xxmWithinMatrix(model = gcsemv, level = "girl", type = "nu", pattern = nu1_pat, value = nu1_val, label = nu1_lab)

**Note**: Observe that the label statement is now included within the ‘theta’ and ‘nu’ matrices. Because the labels used within **boy** and **girl** are identical, these matrices are restricted to be equivalent across the two levels.
Specifying within-level model matrices for 'school' (level 3)

`school` has four latent variables `boywritten`, `boycourse`, `girlwritten`, and `girlcourse`. This level includes a single latent covariance matrix.

1. **psi**: Latent covariance matrix is a (4x4) matrix. The elements include variances of the four latent variables and the six unique covariances between them.

   ```r
gcsemv <- xxmWithinMatrix(model = gcsemv, level = "school", type = "psi", pattern = ps3_pat, value = ps3_val)
   ```

Specifying across-level model matrices connecting 'school' to 'boy' and 'girl'

So far, we have specified within-level matrices. We now need to connect observed and latent variables across-levels. Observed variables `written` and `course` are measured at both of the lowest levels: `boy` and `girl` making a total of four observed variables. Each of these four observed variables has its own latent random-intercept. In other words, a single observed dependent variable within `boy` or `girl` is regressed on a single latent independent variable within `school` with a fixed coefficient of 1.0. This across-level regression connecting variables across two submodels is specified by invoking `xxmBetweenMatrix()`.

**school to boy**

1. **lambda**: Factor-loading matrix is a (2x4) matrix. It has eight different elements, six of which are set to 0 and two of which are set to 1.0.

   ```r
gcsemv <- xxmBetweenMatrix(model = gcsemv, parent = "school", child = "boy", type = "lambda", pattern = ly13_pat, value = ly13_val)
   ```

**school to girl**

1. **lambda**: Factor-loading matrix is a (2x4) matrix. It has eight different elements, six of which are set to 0 and two of which are set to 1.0.

   ```r
gcsemv <- xxmBetweenMatrix(model = gcsemv, parent = "school", child = "girl", type = "lambda", pattern = ly23_pat, value = ly23_val)
   ```

Compute: `xxmRun()`

Model specification is now complete. Parameter estimation is initiated:

```r
gcsemv <- xxmRun(model = gcsemv)
```

Notes

1. Dataset for this model is packaged with `xxm` in an R workspace called `gcsemv.xxm.RData`. The three data frames called 'boy', 'girl', and 'school' are loaded into the R workspace when the library is loaded:

   ```r
   library(xxm)
data(gcsemv.xxm, package="xxm")
   ```

2. Datasets for this model are documented under `gcsemv.xxm.RData` in ???.
3. R script for running this model is stored under .../models/gcsemv.1.xxm.R
4. R script for running the prior model is stored under .../models/gcsemv.0.xxm.R

---

**hcfa.school**

**XXM Dataset: hcfa.school**

---

### Description

hcfa is comprised of 3 simulated datasets: **hcfa.student**, **l2**, and **l3**. The research design corresponds to a common measurement scenario in educational research: a 3 level nested design.

### Usage

```r
data(hcfa.school)
```

### Format

A data frame with 40 observations on the following 4 variables.

- school: a numeric vector
- q1: a numeric vector
- q2: a numeric vector
- q3: a numeric vector

### See Also

hcfa.student; hcfa.teacher

### Examples

```r
data(hcfa.school)
## maybe str(hcfa.school); plot(hcfa.school) ...
```

---

**hcfa.student**

**XXM Dataset: hcfa.student**

---

### Description

hcfa is comprised of 3 simulated datasets: **hcfa.student**, **l2**, and **l3**. The research design corresponds to a common measurement scenario in educational research: a 3 level nested design.

### Usage

```r
data(hcfa.student)
```
hcfa.teacher

Format
A data frame with 600 observations on the following 6 variables.

student a numeric vector
teacher a numeric vector
school a numeric vector
y1 a numeric vector
y2 a numeric vector
y3 a numeric vector

See Also
hcfa.school; hcfa.teacher

Examples
data(hcfa.student)
## maybe str(hcfa.student) ; plot(hcfa.student) ...

hcfa.teacher

XXM Dataset: hcfa.teacher

Description
hcfa is comprised of 3 simulated datasets: hcfa.student, l2, and l3. The research design corresponds to a common measurement scenario in educational research: a 3 level nested design.

Usage
data(hcfa.teacher)

Format
A data frame with 120 observations on the following 2 variables.

teacher a numeric vector
school a numeric vector

See Also
hcfa.student; hcfa.school

Examples
data(hcfa.teacher)
## maybe str(hcfa.teacher) ; plot(hcfa.teacher) ...
Introduction: Hierarchical Confirmatory Factor Analysis

hcfa is a three level hierarchical SEM model with observed dependent variables at levels 1 and 3; and latent variables at all three levels. The model includes a latent variable regression at level 3. The data for this model were simulated. The example is intended to illustrate different within- and between- parameter matrices.

Latent variables at all three levels are defined by observed variables at level-1.

Creating a model: ‘hcfa’

The model involves three levels: student, teacher, school

1. student is nested within both teacher and school. student has a three observed dependent variables: y1, y2, and y3, and a single latent variable A1.

2. teacher is a parent of student. For this model, teacher is not nested under school as the school level random-effect for student latent outcome is modeled directly. teacher has no observed variable and a single latent variable: A2 defined by observed variables at the student level.

3. school is a parent of student. school includes three observed dependent variables: q1, q2, and q3. school submodel has two latent variables: A3 defined by observed variables at the student level and a school level latent variable Q defined by level-3 observed variables.

hcfa is created by invoking xxmModel().

hcfa <- xxmModel(levels = c("student","teacher","school"))

Note: Levels are numbered according to their position in the above command. Hence, student corresponds to level 1, teacher corresponds to level 2, and school corresponds to level 3.

Adding submodels for ‘student’, ‘teacher‘ and ‘school‘

For each level declared above, we need to create corresponding submodels. A submodel is created by invoking xxmSubmodel():

1. student

hcfa <- xxmSubmodel(model = hcfa, level = "student", parents = c("teacher","school"), ys = ys1, xs =, etas = c("A1"), data = hcfa.student)

2. teacher

teacher <- xxmSubmodel(model = hcfa, level="teacher", parents =, ys =, xs =, etas = c("A2"), data = hcfa.teacher)
3. **school**

```r
hcfa <- xxmSubmodel(model = hcfa, level = "school", parents =,
ys = c("q1","q2","q3"), xs =, etas = c("A3","Q3"),
data = hcfa.school)
```

**Specifying within-level model matrices for ‘student’ (level 1)**

**student** model involves a factor-model with all three level-1 observed variables loading on a single latent variable $A_1$. Factor model involves three parameter matrices: factor-loading matrix $\lambda$, latent factor covariance matrix $\psi$, and observed residual-covariance matrix $\theta$. With raw data, we also need to specify measurement intercepts for observed variables using $\nu$.

1. **$\lambda$**: Factor loading matrix is a (3x1) matrix. The first factor loading is fixed to 1.0 and the remaining two are freely estimated.

   ```r
   hcfa <- xxmWithinMatrix(model = hcfa, level = "student",
type = "lambda", pattern = ly1_pat, value = ly1_val)
   ```

2. **$\psi$**: Latent covariance matrix is a (1x1) matrix with freely estimated variance of the latent variable $A_1$.

   ```r
   hcfa <- xxmWithinMatrix(model = hcfa, level = "student",
type = "psi", pattern = ps1_pat, value = ps1_val)
   ```

3. **$\theta$**: Observed residual-covariance matrix is a (3x3) matrix with all three residual variances (diagonal elements) freely estimated.

   ```r
   hcfa <- xxmWithinMatrix(model = hcfa, level = "student",
type = "theta", pattern = th1_pat, value = th1_val)
   ```

4. **$\nu$**: Observed variable intercept matrix is a (3x1) matrix with freely estimated intercepts.

   ```r
   hcfa <- xxmWithinMatrix(model = hcfa, level = "student",
type = "nu", pattern = nu1_pat, value = nu1_val)
   ```

**Specifying within level model matrices for ‘teacher’ (level 2)**

With a single latent variable $A_2$, the submodel for **teacher** is simple. It includes a single variance of the latent dependent variable.

1. **$\psi$**: Latent covariance matrix is a (1x1) matrix with a single element, the variance of the latent variable $A_2$.

   ```r
   hcfa <- xxmWithinMatrix(model = hcfa, level = "teacher",
type = "psi", pattern = ps2_pat, value = ps2_val)
   ```
Specifying within-level model matrices for ‘school’ (level 3)

`school` has two latent variables A3 and Q.

Q is defined by all level-3 observed indicators. As in case of level-1 we need to define a measurement model for Q. Measurement model involves three parameter matrices: factor-loading matrix `lambda`, latent factor covariance matrix `psi`, and observed residual-covariance matrix `theta`. With raw data, we also need to specify measurement intercepts for observed variables using `nu`.

A3 is regressed on Q. The latent regression matrix is called `beta`.

1. `lambda`: Factor loading matrix is a (3x2) matrix. The first latent variable A3 is defined by level-1 indicators. Hence the first column is fixed to 0.0. The second latent variable Q. The first row of the second column is fixed to 1.0 to define the latent variable scale. The last two elements of second column are freely estimated.

   ```
   hcfa <- xxmWithinMatrix(model = hcfa, level = "school",
                          type = "lambda", pattern = ly3_pat, value = ly3_val)
   ```

2. `psi`: Latent covariance matrix is a (2x2) diagonal matrix. The first element is the residual variance of the latent dependent variable A3 and the second element is the unconditional variance of the latent predictor Q.

   ```
   hcfa <- xxmWithinMatrix(model = hcfa, level = "school",
                          type = "psi", pattern = ps2_pat, value = ps2_val)
   ```

3. `theta`: Observed residual-covariance matrix is a (3x3) matrix with all three residual variances (diagonal elements) freely estimated.

   ```
   hcfa <- xxmWithinMatrix(model = hcfa, level = "school",
                          type = "theta", pattern = th3_pat, value = th3_val)
   ```

4. `nu`: Observed variable intercept matrix is a (3x1) matrix with freely estimated intercepts.

   ```
   hcfa <- xxmWithinMatrix(model = hcfa, level = "school",
                          type = "nu", pattern = nu3_pat, value = nu3_val)
   ```

5. `beta`: Latent variable regression matrix is a (2x2) upper-triangular matrix with a single free element \( \beta_{1,2} \).

   ```
   hcfa <- xxmWithinMatrix(model = hcfa, level = "school",
                          type = "beta", pattern = be3_pat, value = be3_val)
   ```

Specifying across-level model matrices connecting ‘teacher’ and ‘student’

So far, we have specified within-level matrices. We now need to connect observed and latent variables across-levels. For this models, latent variables at level-2 and level-3 are measured by observed indicators at level-1. Across-level measurement relationship is specified with a `lambda` matrix. All across-level matrices connecting variables across two submodels are specified by invoking `xxmBetweenMatrix()`.

Note: The direction of influence is from the higher level model `teacher` (2) or the `school` to the lower level model `student` (1) or the `child`. 
1. **lambda**: Factor-loading matrix is a (3x1) matrix. The first element is fixed to 1.0 and the remaining elements are freely estimated.

   ```r
   hcfa <- xxmBetweenMatrix(model = hcfa, parent = "teacher", 
   child = "student", type = "lambda", pattern = ly12_pat, 
   value = ly12_val)
   ```

**Specifying across-level model matrices connecting 'school' and 'teacher'**

The first ILatent variable at leve-3 **A3** is measured by observed indicators at level-1.

1. **lambda**: Factor-loading matrix is a (3x2) matrix. The first latent variable **A3** is defined by level-1 indicators. The first row of the first column is fixed to 1.0 to define the latent variable scale. The last two elements of first column are freely estimated. The second latent variable **Q** was already defined by level-3 observed variables. Hence all elements of second column are fixed to 0.0.

   ```r
   hcfa <- xxmBetweenMatrix(model = hcfa, parent = "school", 
   child = "student", type = "lambda", pattern = ly13_pat, 
   value = ly13_val)
   ```

**Compute: xxmRun()**

Model specification is now complete. Parameter estimation is initiated as follows:

   ```r
   hcfa <- xxmRun(model = hcfa)
   ```

**Notes**

1. Dataset for this model is packaged with xxm in an R workspace called hcfa.xxm.RData. The three data frames called 'hcfa.student', 'hcfa.teacher', and 'hcfa.school' are loaded into the R workspace when the library is loaded:

   ```r
   library(xxm)
   data(hcfa.xxm, package="xxm")
   ```

2. Datasets for this model are documented under hcfa.xxm.RData in ???.

3. R script for running this model is stored under .../models/hcfa.1.xxm.R

**See Also**

hcfa.xxm.2, hcfa.xxm.3.
Introduction: Hierarchical Confirmatory Factor Analysis II

`hcfa` is a three level hierarchical SEM model with observed dependent variables at levels 1 and 3; and latent variables at all three levels. The model includes a latent variable regression at level 3. The data for this model were simulated. The example is intended to illustrate different within- and between- parameter matrices.

Latent variables at all three levels are defined by observed variables at level-1. This model illustrates how equality constraints may be imposed on parameters. In this case, factor loadings for latent variables at all three levels defined by observed variables at level-1 are constrained to be equal. Equality constraints are imposed by using a label matrix. Any two free parameters with the same label are constrained to be equal.

Creating a model: `hcfa`

The model involves three levels: `student`, `teacher`, `school`

1. `student` is nested within both `teacher` and `school`. `student` has three observed dependent variables: `y1`, `y2`, and `y3`, and a single latent variable `A1`.
2. `teacher` is a parent of `student`. For this model, `teacher` is not nested under `school` as the `school` level random-effect for student latent outcome is modeled directly. `teacher` has no observed variable and a single latent variable: `A2` defined by observed variables at the `student` level.
3. `school` is a parent of `student`. `school` includes three observed dependent variables: `q1`, `q2`, and `q3`. `school` submodel has two latent variables: `A3` defined by observed variables at the `student` level and a school level latent variable `Q` defined by level-3 observed variables.

`hcfa` is created by invoking `xxmModel()`.

`hcfa <- xxmModel(levels = c("student","teacher","school"))`

Note: Levels are numbered according to their position in the above command. Hence, `student` corresponds to level 1, `teacher` corresponds to level 2, and `school` corresponds to level 3.

Adding submodels for `student`, `teacher` and `school`

For each level declared above, we need to create corresponding submodels. A submodel is created by invoking `xxmSubmodel()`:

1. `student`

   `hcfa <- xxmSubmodel(model = hcfa, level = "student", parents = c("teacher","school"), ys = ys1, xs =, etas = c("A1"), data = hcfa.student)`

2. `teacher`
teacher <- xxmSubmodel(model = hcfa, level="teacher",
    parents =, ys =, xs =, etas = c("A2"), data = hcfa.teacher)

3. school

hcfa <- xxmSubmodel(model = hcfa, level = "school", parents =,
    ys = c("q1","q2","q3"), xs =, etas = c("A3", "Q3"),
    data = hcfa.school)

Specifying within-level model matrices for ‘student’ (level 1)

student model involves a factor-model with all three level-1 observed variables loading on a single latent variable A1. Factor model involves three parameter matrices: factor-loading matrix lambda, latent factor covariance matrix psi, and observed residual-covariance matrix theta. With raw data, we also need to specify measurement intercepts for observed variables using nu.

1. lambda: Factor loading matrix is a (3x1) matrix. The first factor loading is fixed to 1.0 and the remaining two are freely estimated.

hcfa <- xxmWithinMatrix(model = hcfa, level = "student",
    type = "lambda", pattern = ly1_pat, value = ly1_val, label = ly1_lab)

Note: The above label matrix is constructed as follows:
ly1_lab <- matrix(c("a","b","c"),3,1)

We will use the same three labels for the two across-level factor-loading matrices.

2. psi: Latent covariance matrix is a (1x1) matrix with freely estimated variance of the latent variable A1.

hcfa <- xxmWithinMatrix(model = hcfa, level = "student",
    type = "psi", pattern = ps1_pat, value = ps1_val)

3. theta: Observed residual-covariance matrix is a (3x3) matrix with all three residual variances (diagonal elements) freely estimated.

hcfa <- xxmWithinMatrix(model = hcfa, level = "student",
    type = "theta", pattern = th1_pat, value = th1_val)

4. nu: Observed variable intercept matrix is a (3x1) matrix with freely estimated intercepts.

hcfa <- xxmWithinMatrix(model = hcfa, level = "student",
    type = "nu", pattern = nu1_pat, value = nu1_val)

Specifying within level model matrices for ‘teacher’ (level 2)

With a single latent variable A2, the submodel for teacher is simple. It includes a single variance of the latent dependent variable.

1. psi: Latent covariance matrix is a (1x1) matrix with a single element, the variance of the latent variable A2.

hcfa <- xxmWithinMatrix(model = hcfa, level = "teacher",
    type = "psi", pattern = ps2_pat, value = ps2_val)
Specifying within-level model matrices for ‘school’ (level 3)

`school` has two latent variables A3 and Q.

Q is defined by all level-3 observed indicators. As in case of level-1 we need to define a measurement model for Q. Measurement model involves three parameter matrices: factor-loading matrix `lambda`, latent factor covariance matrix `psi`, and observed residual-covariance matrix `theta`. With raw data, we also need to specify measurement intercepts for observed variables using `nu`.

A3 is regressed on Q. The latent regression matrix is called `beta`.

1. **lambda**: Factor loading matrix is a (3x2) matrix. The first latent variable A3 is defined by level-1 indicators. Hence the first column is fixed to 0.0. The second latent variable Q. The first row of the second column is fixed to 1.0 to define the latent variable scale. The last two elements of second column are freely estimated.

   ```r
   hcfa <- xxmWithinMatrix(model = hcfa, level = "school",
   type = "lambda", pattern = ly3_pat, value = ly3_val)
   ``

2. **psi**: Latent covariance matrix is a (2x2) diagonal matrix. The first element is the residual variance of the latent dependent variable A3 and the second element is the unconditional variance of the latent predictor Q.

   ```r
   hcfa <- xxmWithinMatrix(model = hcfa, level = "school",
   type = "psi", pattern = ps2_pat, value = ps2_val)
   ``

3. **theta**: Observed residual-covariance matrix is a (3x3) matrix with all three residual variances (diagonal elements) freely estimated.

   ```r
   hcfa <- xxmWithinMatrix(model = hcfa, level = "school",
   type = "theta", pattern = th3_pat, value = th3_val)
   ``

4. **nu**: Observed variable intercept matrix is a (3x1) matrix with freely estimated intercepts.

   ```r
   hcfa <- xxmWithinMatrix(model = hcfa, level = "school",
   type = "nu", pattern = nu3_pat, value = nu3_val)
   ``

5. **beta**: Latent variable regression matrix is a (2x2) upper-triangular matrix with a single free element $\beta_{1,2}$.

   ```r
   hcfa <- xxmWithinMatrix(model = hcfa, level = "school",
   type = "beta", pattern = be3_pat, value = be3_val)
   ```

Specifying across-level model matrices connecting ‘teacher‘ and ‘student‘

So far, we have specified within-level matrices. We now need to connect observed and latent variables across-levels. For this models, latent variables at level-2 and level-3 are measured by observed indicators at level-1. Across-level measurement relationship is specified with a `lambda` matrix. All across-level matrices connecting variables across two submodels are specified by invoking `xxmBetweenMatrix()`.

Note: The direction of influence is from the higher level model `teacher` (2) or the `school` to the lower level model `student` (1) or the `child`. 
1. **lambda**: Factor-loading matrix is a (3x1) matrix. The first element is fixed to 1.0 and the remaining elements are freely estimated.

   ```r
   hcfa <- xxmBetweenMatrix(model = hcfa, parent = "teacher", 
                           child = "student", type = "lambda", pattern = ly12_pat, 
                           value = ly12_val, label = ly12_lab)
   ``

   Note: The above label matrix is constructed as follows:

   ```r
   ly12_lab <- matrix(c("a","b","c"),3,1)
   ``

   Labels for the three factor-loadings are identical to those for level-1 factor loading matrix.

**Specifying across-level model matrices connecting ‘school’ and ‘teacher’**

The first latent variable at level-3 A3 is measured by observed indicators at level-1.

1. **lambda**: Factor-loading matrix is a (3x2) matrix. The first latent variable A3 is defined by level-1 indicators. The first row of the first column is fixed to 1.0 to define the latent variable scale. The last two elements of first column are freely estimated. The second latent variable Q was already defined by level-3 observed variables. Hence all elements of second column are fixed to 0.0.

   ```r
   hcfa <- xxmBetweenMatrix(model = hcfa, parent = "school", 
                           child = "student", type = "lambda", pattern = ly13_pat, 
                           value = ly13_val)
   ``

   Note: The above label matrix is constructed as follows:

   ```r
   ly13_lab <- matrix(c("a","b","c","d","e","f"),3,2)
   ``

   Labels for the three factor-loadings are identical to those for level-1 factor loading matrix as well as across-level factor-loadings connecting teacher and student.

**Compute: xxmRun()**

Model specification is now complete. Parameter estimation is initiated as follows:

```r
hcfa <- xxmRun(model = hcfa)
```

**Notes**

1. Dataset for this model is packaged with xxm in an R workspace called hcfa.xxm.RData. The three data frames called 'hcfa.student', 'hcfa.teacher', and 'hcfa.school' are loaded into the R workspace when the library is loaded:

   ```r
   library(xxm) 
data(hcfa.xxm, package="xxm")
   ``

2. Datasets for this model are documented under hcfa.xxm.RData in ???.

3. R script for running this model is stored under .../models/hcfa.2.xxm.R

**See Also**

hcfa.xxm.1, hcfa.xxm.3.
**Introduction: Hierarchical Confirmatory Factor Analysis**

**hcfa** is a three-level hierarchical SEM model with observed dependent variables at levels 1 and 3; and latent variables at all three levels. The model includes a latent variable regression at level 3. The data for this model were simulated. The example is intended to illustrate different within- and between-parameter matrices.

Unlike the previous two models, level-2 and level-3 latent variables are not directly defined by level-1 observed variables. Instead, these variables are defined as "random-intercepts" for level-1 and level-2 latent variables, respectively. Alternatively, the model can also be thought of as a SEM Hierarchical-Factor model of sorts.

Form a practical standpoint there are two differences between the current model and the previous two models:

1. **student** now has a single ‘parent’ **teacher**, and **school** is now a ‘parent’ of **teacher** rather than **school**. This involves changes to the `xxmSubmodel()` statements for **student** and **teacher**.

2. Random-intercept for a latent variables involve across-level regression among latent variables. This model involves across-level latent variable regression. Hence, across-level relationships are defined using **beta** matrices instead of **lambda** matrices.

**Creating a model: ‘hcfa’**

The model involves three levels: **student**, **teacher**, **school**

1. **student** is nested within **teacher** but not **school**. **student** has a three observed dependent variables: \(y_1\), \(y_2\), and \(y_3\); and a single latent variable \(A_1\).

2. **teacher** is a parent of **student** and is itself nested within **school**. **teacher** has no observed variable and a single latent variable: \(A_2\) defined by \(A_1\) at the **student** level.

3. **school** is a parent of **teacher**. **school** includes three observed dependent variables: \(q_1\), \(q_2\), and \(q_3\). **school** submodel has two latent variables: \(A_3\) defined by latent variable \(A_2\) at the **teacher** level and a school level latent variable \(Q\) defined by level-3 observed variables.

**hcfa** is created by invoking `xxmModel()`.

\[
\text{hcfa <- xxmModel(levels = c("student","teacher","school"))}
\]

Note: Levels are numbered according to their position in the above command. Hence, **student** corresponds to level 1, **teacher** corresponds to level 2, and **school** corresponds to level 3.
Adding submodels for ‘student’, ‘teacher’ and ‘school’

For each level declared above, we need to create corresponding submodels. A submodel is created by invoking xxmSubmodel():

1. student

```r
hcfa <- xxmSubmodel(model = hcfa, level = "student",
                    parents = c("teacher"), ys = ys1, xs =,
                    etas = c("A1"), data = hcfa.student)
```

2. teacher

```r
teacher <- xxmSubmodel(model = hcfa, level="teacher",
                    parents = c("school"), ys =, xs =, etas = c("A2"), data = hcfa.teacher)
```

3. school

```r
hcfa <- xxmSubmodel(model = hcfa,level = "school", parents =,
                    ys = c("q1","q2","q3"), xs =, etas = c("A3", "Q3"),
                    data = hcfa.school)
```

Specifying within-level model matrices for ‘student’ (level 1)

The **student** model involves a factor-model with all three level-1 observed variables loading on a single latent variable $A_1$. Factor model involves three parameter matrices: factor-loading matrix $\lambda$, latent factor covariance matrix $\psi$, and observed residual-covariance matrix $\theta$. With raw data, we also need to specify measurement intercepts for observed variables using $\nu$.

1. **lambda**: Factor loading matrix is a $(3 \times 1)$ matrix. The first factor loading is fixed to 1.0 and the remaining two are freely estimated.

```r
hcfa <- xxmWithinMatrix(model = hcfa, level = "student",
                        type = "lambda", pattern = ly1_pat, value = ly1_val)
```

2. **psi**: Latent covariance matrix is a $(1 \times 1)$ matrix with freely estimated variance of the latent variable $A_1$.

```r
hcfa <- xxmWithinMatrix(model = hcfa, level="student",
                        type = "psi", pattern = ps1_pat, value = ps1_val)
```

3. **theta**: Observed residual-covariance matrix is a $(3 \times 3)$ matrix with all three residual variances (diagonal elements) freely estimated.

```r
hcfa <- xxmWithinMatrix(model = hcfa, level = "student",
                        type = "theta", pattern = th1_pat, value = th1_val)
```

4. **nu**: Observed variable intercept matrix is a $(3 \times 1)$ matrix with freely estimated intercepts.

```r
hcfa <- xxmWithinMatrix(model = hcfa, level = "student",
                        type = "nu", pattern = nu1_pat, value = nu1_val)
```
Specifying within level model matrices for ‘teacher’ (level 2)

With a single latent variable \( A_2 \), the submodel for teacher is simple. It includes a single variance of the latent dependent variable.

1. \( \psi \): Latent covariance matrix is a (1x1) matrix with a single element, the variance of the latent variable \( A_2 \).

\[
\text{hcfa} <- \text{xxmWithinMatrix(model = hcfa, level = "teacher", type = "psi", pattern = ps2_pat, value = ps2_val)}
\]

Specifying within-level model matrices for ‘school’ (level 3)

school has two latent variables \( A_3 \) and \( Q \).

\( Q \) is defined by all level-3 observed indicators. As in case of level-1 we need to define a measurement model for \( Q \). Measurement model involves three parameter matrices: factor-loading matrix \( \lambda \), latent factor covariance matrix \( \psi \), and observed residual-covariance matrix \( \theta \). With raw data, we also need to specify measurement intercepts for observed variables using \( \nu \).

\( A_3 \) is regressed on \( Q \). The latent regression matrix is called \( \beta \).

1. \( \lambda \): Factor loading matrix is a (3x2) matrix. The first latent variable \( A_3 \) is defined by level-1 indicators. Hence the first column is fixed to 0.0. The second latent variable \( Q \). The first row of the second column is fixed to 1.0 to define the latent variable scale. The last two elements of second column are freely estimated.

\[
\text{hcfa} <- \text{xxmWithinMatrix(model = hcfa, level = "school", type = "lambda", pattern = ly3_pat, value = ly3_val)}
\]

2. \( \psi \): Latent covariance matrix is a (2x2) diagonal matrix. The first element is the residual variance of the latent dependent variable \( A_3 \) and the second element is the unconditional variance of the latent predictor \( Q \).

\[
\text{hcfa} <- \text{xxmWithinMatrix(model = hcfa, level = "school", type = "psi", pattern = ps2_pat, value = ps2_val)}
\]

3. \( \theta \): Observed residual-covariance matrix is a (3x3) matrix with all three residual variances (diagonal elements) freely estimated.

\[
\text{hcfa} <- \text{xxmWithinMatrix(model = hcfa, level = "school", type = "theta", pattern = th3_pat, value = th3_val)}
\]

4. \( \nu \): Observed variable intercept matrix is a (3x1) matrix with freely estimated intercepts.

\[
\text{hcfa} <- \text{xxmWithinMatrix(model = hcfa, level = "school", type = "nu", pattern = nu3_pat, value = nu3_val)}
\]

5. \( \beta \): Latent variable regression matrix is a (2x2) upper-triangular matrix with a single free element \( \beta_{1,2} \).

\[
\text{hcfa} <- \text{xxmWithinMatrix(model = hcfa, level = "school", type = "beta", pattern = be3_pat, value = be3_val)}
\]
Specifying across-level model matrices connecting ‘teacher’ and ‘student’

So far, we have specified within-level matrices. We now need to connect observed and latent variables across-levels. For this models, latent variables at level-2 and level-3 are measured by level-1 latent variable A1. Such across-level structural relationship is specified with a beta matrix. All across-level matrices connecting variables across two submodels are specified by invoking xxmBetweenMatrix().

1. **beta**: Latent variable regression matrix is a (1x1) matrix. The single element is fixed to 1.0 suggesting that the level-2 latent variable A2 is a random-intercept for the level-1 latent variable A1.

   \[
   \begin{align*}
   \text{hcfa} & \leftarrow \text{xxmBetweenMatrix(model = hcfa, parent = "teacher", child = "student", type = "beta", pattern = be12_pat, value = be12_val)}
   \end{align*}
   \]

Specifying across-level model matrices connecting ‘school’ and ‘teacher’

The first latent variable at leve-3 A3 is measured by level-2 latent variable A2.

1. **beta**: Factor-loading matrix is a (1x2) matrix. The first latent variable A3 is defined by level-2 latent variable A2 and is fixed to 1.0. The second latent variable Q was already defined by level-3 observed variables. Hence the second element is fixed to 0.0.

   \[
   \begin{align*}
   \text{hcfa} & \leftarrow \text{xxmBetweenMatrix(model = hcfa, parent = "school", child = "teacher", type = "beta", pattern = be23_pat, value = be23_val)}
   \end{align*}
   \]

Compute: xxmRun()

Model specification is now complete. Parameter estimation is initiated as follows:

   \[
   \text{hcfa} \leftarrow \text{xxmRun(model = hcfa)}
   \]

Notes

1. Dataset for this model is packaged with xxm in an R workspace called hcfa.xxm.RData. The three data frames called ‘hcfa.student’, ‘hcfa.teacher’, and ‘hcfa.school’ are loaded into the R workspace when the library is loaded:

   \[
   \begin{align*}
   \text{library(xxm)}
   \\
   \text{data(hcfa.xxm, package="xxm")}
   \end{align*}
   \]

2. Datasets for this model are documented under hcfa.xxm.RData in ???.

3. R script for running this model is stored under …/models/hcfa.3.xxm.R

See Also

hcfa.xxm.1, hcfa.xxm.2.
What is a level?

Description

Description of levels in xxm.

Details

Notion of a level is central to multi-level models. The idea is even more important for specifying an xxm model. In xxm, a level is defined as an abstract entity for which there are multiple exchangeable units. Typically, a level may have one or more observed and/or latent variables.

Levels may be related to one another in a hierarchical fashion. For example, if students are nested within teachers, the student level is said to be the child level and teacher level is the corresponding parent level. Parent and child levels not only indicate a hierarchical nesting of students within teachers, but is also intended to identify the direction of influence. Teacher level will have one or more latent variables and/or exogenous predictors that are predictors of student level observed or latent variables. In this case, student level is said to be at a lower level than the teacher level.

The notion of parent and child levels applies to models with more than two levels. For example, a three-level hierarchical model may include student, classroom, school as levels. However, there are two possibilities with regards to parent-child relationship among the three levels.

(1) Classroom and school may both be parents of student.

(2) Classroom is a parent of student and school is a parent of classroom.

The distinction between the two specification lies in having observed/latent variables at the parent level that influence observed/latent variables at the child level. In the second instance, school does not directly influence the student level. Instead, its influence may be mediated via latent variables at the classroom level. It is possible to specify a mathematically identical model using either of these formulations.

Note

(1) Levels to be included in the model are specified in xxmModel command with the levels argument. Here levels must be ordered from lower to higher numbers i.e., bottom-up.

(2) Each level listed in xxmModel must have corresponding level-specific model specified using xxmSubmodel.

(3) Parent-child relationship across any two levels is specified using parents argument of xxmSubmodel command.

(4) For any two levels in a parent-child relationship, a corresponding regression relationship must be defined across the two levels.

See Also

xxmModel, xxmSubmodel
**Description**

`lranslp` contains two data frames: `lranslp.l1` and `lranslp.l2`, that have been formatted for xxM analysis. These data were simulated to reflect a common research design in educational settings: two-level latent random intercept regressed on an observed level predictor, with random slopes.

**Usage**

```r
data(lranslp.l1)
```

**Format**

A data frame with 900 observations on the following 7 variables.

- `l1`: a numeric vector
- `l2`: a numeric vector
- `y1`: a numeric vector
- `y2`: a numeric vector
- `y3`: a numeric vector
- `y4`: a numeric vector
- `x`: a numeric vector

**See Also**

`lranslp.l2`

**Examples**

```r
data(lranslp.l1)
## maybe str(lranslp.l1); plot(lranslp.l1) ...
```

---

**Description**

`lranslp` contains two data frames: `lranslp.l1` and `lranslp.l2`, that have been formatted for xxM analysis. These data were simulated to reflect a common research design in educational settings: two-level latent random intercept regressed on an observed level predictor, with random slopes.
Usage

data(ltranslp.l2)

Format

A data frame with 150 observations on the following variable.

l12  a numeric vector

See Also

ltranslp.l1

Examples

data(ltranslp.l2)
## maybe str(ltranslp.l2) ; plot(ltranslp.l2) ...

<table>
<thead>
<tr>
<th>lwid.first</th>
<th>XXM Dataset: lwid.first</th>
</tr>
</thead>
</table>

Description

lwid consists of longitudinal student reading data (letter word identification) for three grades (kinder, first and second). Students are nested within different teachers in each grade. Students are also nested within schools. There are six datasets response, kinder, first, second, student, and school.

Usage

data(lwid.first)

Format

A data frame with 154 observations on the following variable.

first  a numeric vector

Examples

data(lwid.first)
## maybe str(lwid.first) ; plot(lwid.first) ...
**Description**

lwid consists of longitudinal student reading data (letter word identification) for three grades (kinder, first and second). Students are nested within different teachers in each grade. Students are also nested within schools. There are six datasets `response`, `kinder`, `first`, `second`, `student`, and `school`.

**Usage**

data(lwid.kinder)

**Format**

A data frame with 87 observations on the following variable.

- `kinder` a numeric vector

**Examples**

data(lwid.kinder)

## maybe str(lwid.kinder) ; plot(lwid.kinder) ...

**Description**

lwid consists of longitudinal student reading data (letter word identification) for three grades (kinder, first and second). Students are nested within different teachers in each grade. Students are also nested within schools. There are six datasets `response`, `kinder`, `first`, `second`, `student`, and `school`.

**Usage**

data(lwid.response)

**Format**

A data frame with 6969 observations on the following 16 variables.

- `response` a numeric vector
- `kinder` a numeric vector
- `first` a numeric vector
- `second` a numeric vector
- `student` a numeric vector
school  a numeric vector
gk1   a numeric vector
gk2   a numeric vector
g11   a numeric vector
g12   a numeric vector
g21   a numeric vector
g22   a numeric vector
LWE   a numeric vector
int   a numeric vector
cent  a numeric vector
quad  a numeric vector

Examples

data(lwid.response)
## maybe str(lwid.response) ; plot(lwid.response) ...

lwid.school  XXM Dataset: lwid.school

Description

lwid consists of longitudinal student reading data (letter word identification) for three grades (kinder, first and second). Students are nested within different teachers in each grade. Students are also nested within schools. There are six datasets response, kinder, first, second, student, and school.

Usage

data(lwid.school)

Format

A data frame with 33 observations on the following variable.

school  a numeric vector

Examples

data(lwid.school)
## maybe str(lwid.school) ; plot(lwid.school) ...
lwid.second  

**Description**

lwid consists of longitudinal student reading data (letter word identification) for three grades (kinder, first and second). Students are nested within different teachers in each grade. Students are also nested within schools. There are six datasets `response`, `kinder`, `first`, `second`, `student`, and `school`.

**Usage**

```r
data(lwid.second)
```

**Format**

A data frame with 150 observations on the following variable.

- `second` a numeric vector

**Examples**

```r
data(lwid.second)
## maybe str(lwid.second) ; plot(lwid.second) ...
```

lwid.student  

**Description**

lwid consists of longitudinal student reading data (letter word identification) for three grades (kinder, first and second). Students are nested within different teachers in each grade. Students are also nested within schools. There are six datasets `response`, `kinder`, `first`, `second`, `student`, and `school`.

**Usage**

```r
data(lwid.student)
```

**Format**

A data frame with 1858 observations on the following variable.

- `student` a numeric vector

**Examples**

```r
data(lwid.student)
## maybe str(lwid.student) ; plot(lwid.student) ...
```
mlcfa contains two data frames: \texttt{mlcfa.student} and \texttt{mlcfa.teacher}, that have been formatted for xxM analysis. These data were simulated to reflect a common research design in educational research: a 2-level confirmatory factor analysis.

Usage

```r
data(mlcfa.student)
```

Format

A data frame with 1141 observations on the following 6 variables.

- \texttt{student} a numeric vector
- \texttt{teacher} a numeric vector
- \texttt{LCE} a numeric vector
- \texttt{PCE} a numeric vector
- \texttt{PVE} a numeric vector
- \texttt{VAE} a numeric vector

See Also

\texttt{mlcfa.teacher}

Examples

```r
data(mlcfa.student)
## maybe str(mlcfa.student) ; plot(mlcfa.student) ... 
```

mlcfa contains two data frames: \texttt{mlcfa.student} and \texttt{mlcfa.teacher}, that have been formatted for xxM analysis. These data were simulated to reflect a common research design in educational research: a 2-level confirmatory factor analysis.

Usage

```r
data(mlcfa.teacher)
```
pcwa.student

**Format**
A data frame with 163 observations on the following variable.

- `teacher` a numeric vector

**See Also**
`mlcfa.student`

**Examples**
```r
data(mlcfa.teacher)
## maybe str(mlcfa.teacher) ; plot(mlcfa.teacher) ...
```

---

**pcwa.student**

**Description**

*pcwa* involves a two level hierarchical linear model with passage comprehension as the dependent variable and word attack as the independent variable.

**Usage**
```
data(pcwa.student)
```

**Format**
A data frame with 802 observations on the following 4 variables.

- `student` a numeric vector
- `teacher` a numeric vector
- `pc` a numeric vector
- `wa` a numeric vector

**Source**
The model and analysis are presented in a book chapter by Lee Alan Martin....

**References**
The model and analysis are presented in a book chapter by Lee Alan Martin....

**See Also**
`pcwa.teacher`
Examples

```r
data(pcwa.student)
## maybe str(pcwa.student) ; plot(pcwa.student) ...
```

`pcwa.teacher`  
**XXM DATASET: pcwa.teacher**

Description

`pcwa` involves a two level hierarchical linear model with passage comprehension as the dependent variable and word attack as the independent variable.

Usage

```r
data(pcwa.teacher)
```

Format

A data frame with 93 observations on the following variable.

- `teacher`  a numeric vector

See Also

`pcwa.student`

Examples

```r
data(pcwa.teacher)
## maybe str(pcwa.teacher) ; plot(pcwa.teacher) ...
```

`pcwa.xxm.1`  
**xxM Model Example: pcwa.xxm.1.R**

Introduction

`pcwa` is a 2-level random intercept model of English reading passage comprehension for 802 students in 92 classrooms at the end of first grade. The goal of the analysis is to partition variability in reading comprehension between classrooms (level-2) and students (level-1). This is the first xxM model (called "EngPC") of the chapter:


This example illustrates how a random-intercept model may be estimated. Random-intercept at level-2 is just a latent variable measured by observed student scores with a fixed factor loading of 1.0.
Creating a model: ‘pcwa’

The model involves two levels: student and teacher

1. **student** is nested within **teacher**. **student** is said to be the child level with **teacher** as the parent level.  
   **student** has 4 variables: **student**, **teacher**, **pc**, **wa**. **student** and **teacher** are the ID variables.  
   **pc** is the single dependent variable. This example does not include any predictors.

2. **teacher** has a single variable: **teacher** to indicate a classroom for each student.  
   While there are no classroom level predictors at the teacher level, the teacher IDs for this simple random intercepts model identify the second level units for which we wish to estimate variability separately from students.

**pcwa** is created by invoking `xxmModel()`.

```r
pcwa <- xxmModel(levels = c("student","teacher"))
```

Adding submodels for ‘student’ and ‘teacher’

For each level declared above, we need to create corresponding submodels. A submodel is created by invoking `xxmSubmodel()`:

1. **student**

   ```r
   pcwa <- xxmSubmodel(model = pcwa, level = "student", parents = c("teacher"), 
                        ys = "pc", xs = , etas = , data = pcwa.student)
   ```

2. **teacher**

   ```r
   pcwa <- xxmSubmodel(model = pcwa, level = "teacher", parents = , 
                        ys = , xs = , etas = c("int"), data = pcwa.teacher)
   ```

Specifying within-level model matrices for ‘student’ (level 1)

With a single dependent variable **pc**, the submodel for **student** includes the residual variance of the dependent variable. Model matrices for each submodel is specified by invoking `xxmWithinMatrix()`:

1. **theta**: Observed residual covariance matrix is a (1x1) matrix with a single element, the residual variance.

   ```r
   pcwa <- xxmWithinMatrix(model = pcwa, level = "student", type = "theta", 
                            pattern = theta_pattern, value = theta_value)
   ```

Specifying within level model matrices for ‘teacher’ (level 2)

With a single latent variable **int**, the submodel for **teacher** includes a single variance and mean of the latent dependent variable.

1. **psi**: Latent covariance matrix is a (1x1) matrix with a single element, the variance of the random intercept **int**.

   ```r
   pcwa <- xxmWithinMatrix(model = pcwa, level = "teacher", type = "psi", 
                           pattern = psi_pattern, value = psi_value)
   ```
2. **alpha**: Latent mean structure is a (1x1) matrix with a single element, the mean of the random intercept `int`.

```r
pcwa <- xxmWithinMatrix(model = pcwa, level = "teacher", type = "alpha", pattern = alpha_pattern, value = alpha_value)
```

**Specifying across-level model matrices connecting ‘student’ and ‘teacher’**

So far, we have specified within-level matrices. We now need to connect observed and latent variables across levels. Observed variable `pc` measured at the lowest level **student** has a latent random intercept `int` at the **teacher** level. In other words, a single observed dependent variable within `pc` is regressed on a single latent independent variable `int` with a fixed coefficient of 1.0. This across-level regression connecting variables across two submodels is specified by invoking `xxmBetweenMatrix()`.

Note: The direction of influence is from the higher level model **teacher** (2) or the **parent** to the lower level model **student** (1) or the **child**.

1. **lambda**: Factor-loading matrix is a (1x1) matrix. It has a single element fixed to 1.0 reflecting the fact that the `int` is defined by `pc`.

```r
pcwa <- xxmBetweenMatrix(model = pcwa, parent = "teacher", child = "student", type = "lambda", pattern = lambda_pattern, value = lambda_value)
```

**Compute: xxmRun()**

Model specification is now complete. Parameter estimation is initiated:

```r
pcwa <- xxmRun(model = pcwa)
```

**Notes**

1. Dataset for this model is packaged with xxm in an R workspace called `pcwa.xxm.RData`. The two data frames called ‘student’ and ‘teacher’ are loaded into the R workspace when the library is loaded:
   ```r
   library(xxm)
   data(pcwa.xxm, package="xxm")
   ```

2. Datasets for this model are documented under `pcwa.xxm.RData` in ???.

3. R script for running this model is stored under `.../models/pcwa.xxm.1.R`
Introduction

**rolemodel** is a three level SEM model for complex teams. An individual may play the role of a team-member in several different teams. An individual may also play the role of a team-leader in multiple different teams. The sole observed outcome variable is an individual’s "performance" tied to a specific team. Clearly, ‘performance’ is cross-classified within individuals and teams. The main issue with such a conceptualization is that an individual may play the role of a team-member in one team and a team-leader in another. Hence, individuals and teams are not independent as in ordinary cross-classified models. Unlike traditional team structure with a single team-leader per team, the current data-structure involves nesting of teams within team-leaders. Hence, we need to account for the possible effect of team-leaders on team-performance. We also need to consider the possibility that an individual’s ‘true’ performance relates to their ability as a team-leader and hence their influence on team-performance.

Creating a model: ‘rolemodel’

The model involves three levels: response, team, individual

1. **response** is nested within both team and individual. response is said to be the child level with team and individual as the corresponding parent levels. response has a single observed dependent variable: performance.
2. **team** is partially nested within individual and is a parent of response. team has a single latent variable: team-performance. In the parlance of multilevel modeling, team-performance is the random-intercept of individual performance at the team level. team has no observed variables.
3. **individual** is not nested within any level and is a parent of both response and team.

It seems unusual for a team to be nested within an individual. Afterall, team members are nested within teams. In this case, only those individuals that play the role of a team-leader will have a link between their individual model with the corresponding team model. In other words, team is really nested within a ‘virtual role’ called leader that ‘real’ individuals play. With this additional notion of a virtual role and a corresponding role-model, nesting of teams within team-leaders makes more sense. xxm figures out the intricacies of connecting only team-leaders and not team-members with their respective teams behind the scene.

individual has two latent variables:

(a) **individual-performance** is the random intercept of response performance at the individual level across all teams that they are a member of.

(b) **leader-performance** is the random intercept of team-performance that an individual happens to lead.

**rolemodel** is created by invoking xxmModel().

rolemodel <- xxmModel(levels = c("response","team","individual"))

Note: Levels are numbered according to their position in the above command. Hence, response corresponds to level 1, team corresponds to level 2, and individual corresponds to level 3.
Adding submodels for ‘response’, ‘team’ and ‘individual’

For each level declared above, we need to create corresponding submodels. A submodel is created by invoking `xxmSubmodel()`:

1. **response**

   ```r
   rolemodel <- xxmSubmodel(model = rolemodel, level = "response",
                            parents = c("team","individual"), ys = "performance",
                            xs =, etas =, data = rolemodels.response)
   ``

2. **team**

   ```r
   rolemodel<- xxmSubmodel(model = rolemodel, level = "team", parents = "individual",
                            ys =, xs =, etas = "perf_team", data = rolemodels.team)
   ``

3. **individual**

   ```r
   rolemodel <- xxmSubmodel(model = rolemodel, level = "individual", parents =, ys =,
                            xs =, etas = c("perf_individual", "perf_leader"), data = rolemodels.individual)
   ``

Specifying within-level model matrices for ‘response’ (level 1)

With a single dependent variable **performance**, the submodel for **response** is simple. It includes the residual variance and the intercept of the dependent variable. Model matrices for each submodel is specified by invoking `xxmWithinMatrix()`:

1. **theta**: Observed residual-covariance matrix is a (1x1) matrix with a single element, the residual variance.

   ```r
   rolemodel <- xxmWithinMatrix(model = rolemodel, level = "response",
                                type = "theta", pattern = th1_pat, value = th1_val)
   ``

2. **nu**: Observed variable intercept matrix is a (1x1) matrix with a single element, the intercept.

   ```r
   rolemodel <- xxmWithinMatrix(model = rolemodel, level = "response",
                                type = "nu", pattern = nu1_pat, value = nu1_val)
   ``

Specifying within level model matrices for ‘team’ (level 2)

With a single latent variable **team-performance**, the submodel for **team** is also simple. It includes a single residual variance of the latent dependent variable.

1. **psi**: Latent residual-covariance matrix is a (1x1) matrix with a single element, the residual variance of the random-intercept **team-performance**.

   ```r
   rolemodel <- xxmWithinMatrix(model = rolemodel, level = "team",
                                type = "psi", pattern = ps2_pat, value = ps2_val)
   ```
Specifying within-level model matrices for ‘individual’ (level 3)

*individual* has two latent variables *individual-performance* and *leader-performance*. The level includes a single latent covariance matrix.

1. **psi**: Latent covariance matrix is a (2x2) matrix. The elements include variances of the two latent variables: *individual-performance* and *leader-performance* and their covariance.

   rolemodel <- xxmWithinMatrix(model = rolemodel, level = "individual", type = "psi", pattern = ps3_pat, value = ps3_val)

Specifying across-level model matrices connecting ‘team’ and ‘response’

So far, we have specified within-level matrices. We now need to connect observed and latent variables across-levels. Observed variable *performance* measured at the lowest level *response* has a latent random-intercept *team-performance* at the *team* level. In other words, a single observed dependent variable within *response* is regressed on a single latent independent variable *team-performance* with a fixed coefficient of 1.0. This across-level regression connecting variables across two submodels is specified by invoking `xxmBetweenMatrix()`.

Note: The direction of influence is from the higher level model *team* (2) or the parent to the lower level model *response* (1) or the child.

1. **lambda**: Factor-loading matrix is a (1x1) matrix. It has a single element fixed to 1.0 reflecting the fact that the *team-performance* is defined by *performance*.

   rolemodel <- xxmBetweenMatrix(model = rolemodel, parent = "team", child = "response", type = "lambda", pattern = ly12_pat, value = ly12_val)

Specifying across-level model matrices connecting ‘individual’ and ‘response’

Observed variable *performance* measured at the lowest level *response* has a latent random-intercept *individual-performance* at the *individual* level. *individual-performance* In other words, a single observed dependent variable within *response* is regressed on a single latent independent variable *team-performance* with a fixed coefficient of 1.0. This across-level regression connecting variables across two submodels is specified by invoking `xxmBetweenMatrix()`.

Note: The direction of influence is from the higher level model *individual* (3) or the parent to the lower level model *response* (1) or the child.

1. **lambda**: Factor-loading matrix is a (1x2) matrix. This is because *response* has a single observed dependent variable and *individual* has two latent variables. The first latent variable *individual-performance* is the random-intercept of *performance*. Hence, the first element is fixed to 1.0. The second latent variable *leader-performance* is defined by *team*. Hence, the second element of the lambda matrix is fixed to 0.0.

   rolemodel <- xxmBetweenMatrix(model = response, parent = "individual", child = "response", type = "lambda", pattern = ly13_pat, value = ly13_val)
Specifying across-level model matrices connecting ‘team’ and ‘individual’

Latent variable team-performance has a latent random-intercept leader-performance at the individual level. In other words, a single latent dependent variable at team is regressed on a single latent independent variable leader-performance with a fixed coefficient of 1.0.

Note: The direction of influence is from the higher level model individual (3) or the parent to the lower level model team (2) or the child.

1. beta: Latent-variable regression matrix (beta) is a (1x2) matrix. This is because team has a single latent dependent variable and individual has two latent variables. The first latent variable individual-performance is the random-intercept of performance. Hence, the first element is fixed to 0.0. The second latent variable leader-performance is defined by team-performance. Hence, the second element of the lambda matrix is fixed to 1.0.

\[
\text{model} \leftarrow \text{xxmBetweenMatrix}(\text{model} = \text{rolemodel}, \text{parent} = \text{"individual"}, \text{child} = \text{"team"}, \text{type} = \text{"beta"}, \text{pattern} = \text{be23\_pat}, \text{value} = \text{be23\_val})
\]

Compute: xxmRun()

Model specification is now complete. Parameter estimation is initiated:

\[
\text{rolemodel} \leftarrow \text{xxmRun}(\text{model} = \text{rolemodel})
\]

Notes

1. Dataset for this model is packaged with xxm in an R workspace called rolemodel.xxm.RData. The three data frames called ‘response’, ‘team’, and ‘individual’ are loaded into the R workspace when the library is loaded:

\[
\text{library(xxm)}
\]
\[
\text{data(\text{rolemodel.xxm, package=\"xxm\")}
\]

2. Datasets for this model are documented under rolemodel.xxm.RData in ???.

3. R script for running this model is stored under \text{.../models/rolemodel.1.xxm.R}

rolemodels.individual  \text{XXM Dataset: rolemodels.individual}

Description

rolemodels contains measures of team member performance. Teams are comprised of individual team members. Each team has a team leader. Each individual may be a member of multiple teams. An individual may also lead multiple different teams.

The data for fitting xxm model comprises of three related data-frames: rolemodels.response, rolemodels.individual, rolemodels.team

Usage

\[
\text{data(\text{rolemodels.individual})}
\]
rolemodels.response

Format
A data frame with 50 observations on the following variable.

individual  a numeric vector

Examples

data(rolemodels.individual)
## maybe str(rolemodels.individual) ; plot(rolemodels.individual) ...
rolemodels.team  

**Description**

`rolemodels` contains measures of team member performance. Teams are comprised of individual team members. Each team has a team leader. Each individual may be a member of multiple teams. An individual may also lead multiple different teams.

The data for fitting xxm model comprises of three related data-frames: `rolemodels.response`, `rolemodels.individual`, `rolemodels.team`

**Usage**

```r
data(rolemodels.team)
```

**Format**

A data frame with 80 observations on the following 2 variables.

- `team` a numeric vector
- `individual` a numeric vector

**Examples**

```r
data(rolemodels.team)
## maybe str(rolemodels.team) ; plot(rolemodels.team) ...
```

----------

ScotsSec.primary  

**Description**

ScotsSec contains three data frames: `ScotsSec.student`, `ScotsSec.primary` and `ScotsSec.secondary`, that have been formatted for xxM analysis. These data were drawn from an example found in Goldstein (1995, 2003) and features a sample of 3435 school children in Fife, Scotland (Paterson, 1991).

**Usage**

```r
data(ScotsSec.primary)
```

**Format**

A data frame with 148 observations on the following variable.

- `primary` a numeric vector
ScotsSec.secondary

See Also

ScotsSec.student; ScotsSec.secondary

Examples

data(ScotsSec.primary)
## maybe str(ScotsSec.primary) ; plot(ScotsSec.primary) ...

ScotsSec.secondary  XXM Dataset: ScotsSec.secondary

Description

ScotsSec contains three data frames: ScotsSec.student, ScotsSec.primary and ScotsSec.secondary, that have been formatted for xxM analysis. These data were drawn from an example found in Goldstein (1995, 2003) and features a sample of 3435 school children in Fife, Scotland (Paterson, 1991).

Usage

data(ScotsSec.secondary)

Format

A data frame with 19 observations on the following variable.

  secondary  a numeric vector

See Also

ScotsSec.student; ScotsSec.primary

Examples

data(ScotsSec.secondary)
## maybe str(ScotsSec.secondary) ; plot(ScotsSec.secondary) ...
Description

ScotsSec contains three data frames: ScotsSec.student, ScotsSec.primary and ScotsSec.secondary, that have been formatted for xxM analysis. These data were drawn from an example found in Goldstein (1995, 2003) and features a sample of 3435 school children in Fife, Scotland (Paterson, 1991).

Usage

data(ScotsSec.student)

Format

A data frame with 3435 observations on the following 4 variables:

- student a numeric vector
- primary a numeric vector
- secondary a numeric vector
- reading a numeric vector

See Also

ScotsSec.secondary; ScotsSec.primary

Examples

data(ScotsSec.student)
## maybe str(ScotsSec.student) ; plot(ScotsSec.student) ...
Creating a model: ‘ScotsSec’

The model involves three levels: **student, primary, secondary**

1. **student** is nested within **primary** and **secondary**.
2. **primary** is not nested within any level and has a single latent variable. **primary_reading** is the random intercept of response **reading** at the **student** level.
3. **secondary** is not nested within any level and has a single latent variable. **secondary_reading** is the random intercept of **student** level outcome **reading**.

The model is created by invoking `xxmModel()`.
```r
ss <- xxmModel(levels = c("student","primary","secondary"))
```

**Note:** Levels are numbered according to their position in the above command. Hence, **student** corresponds to level 1, **primary** corresponds to level 2, and **secondary** corresponds to level 3.

Adding submodels for ‘student’, ‘primary’ and ‘secondary’

For each level declared above, we need to create corresponding submodels. Each submodel contains all the elements that will exist within a level. A submodel is created by invoking `xxmSubmodel()`:

1. **student**
   ```r
   ss <- xxmSubmodel(model = ss, level = "student", parents = c("primary","secondary"), ys = "reading")
   ```
2. **primary**
   ```r
   ss <- xxmSubmodel(model = ss, level = "primary", parents = , ys = , xs = , etas = "primary_reading", data = ScotsSec.primary)
   ```
3. **secondary**
   ```r
   ss <- xxmSubmodel(model = ss, level = "secondary", parents = , ys = , xs = , etas = "secondary_reading", data = ScotsSec.secondary)
   ```

Specifying within-level model matrices for ‘student’ (level 1)

The student submodel has only one observed variables (**reading**). Model matrices for each submodel is specified by invoking `xxmWithinMatrix()`:

1. **theta**: Observed residual-covariance matrix is a (1x1) matrix with a single element, the residual variance of **reading**.
   ```r
   ss <- xxmWithinMatrix(model = ss, level = "student", type = "theta", pattern = th1_pat, value = th1_val)
   ```
2. **nu**: Observed variable intercept matrix is a (1x1) matrix with one element, the intercept of **reading**
   ```r
   ss <- xxmWithinMatrix(model = ss, level = "student", type = "nu", pattern = nu1_pat, value = nu1_val)
   ```

Specifying within-level model matrices for ‘primary’ (level 2)

The **primary** level has one latent variable (**primary_reading**). Model matrices for each submodel is specified by invoking `xxmWithinMatrix()`:

1. **psi**: Latent residual-covariance matrix is a (1x1) matrix with a single element, the variance of **primary_reading**.
   ```r
   ss <- xxmWithinMatrix(model = ss, level = "primary", type = "psi", pattern = ps2_pat, value = ps2_val)
   ```
Specifying within-level model matrices for ‘secondary’ (level 3)

*secondary* has one latent variable: *secondary_reading*. This level includes a single latent covariance matrix.

1. **psi**: Latent covariance matrix is a (1x1) matrix. Which consists of the variance of the latent variable *secondary_reading*.
   
   ```r
   ss <- xxmWithinMatrix(model = ss, level = "secondary", type = "psi", pattern = ps3_pat, value = ps3_val)
   ``

Specifying across-level model matrices connecting 'student' and 'primary' to 'secondary'

So far, we have specified within-level matrices. We now need to connect observed and latent variables across-levels. In each of the connections, there is one observed variable that is an indicator for a single latent variable. *xxm* will match the id variables in *student* to the proper upper level. These across-level regression connecting variables across two submodels is specified by invoking *xxmBetweenMatrix()*. 

Specifying across-level model matrices connecting primary to student

1. **lambda**: Factor-loading matrix is a (1x1) matrix. A single element restricted to 1.0.
   
   ```r
   ss <- xxmBetweenMatrix(model = ss, parent = "secondary", child = "student", type = "lambda", pattern = ly12_pat, value = ly12_val)
   ``

Specifying across-level model matrices connecting secondary to student

1. **lambda**: Factor-loading matrix is a (1x1) matrix. A single element restricted to 1.0.
   
   ```r
   ss <- xxmBetweenMatrix(model = ss, parent = "secondary", child = "primary", type = "lambda", pattern = ly13_pat, value = ly13_val)
   ``

**Compute: xxmRun()**

Model specification is now complete. Parameter estimation can now begin:

```r
ss <- xxmRun(model = ss)
```

**Notes**

1. Dataset for this model is packaged with *xxm* in an R workspace called *ScotsSec.xxm.RData*. The three data frames called 'student', 'primary', and 'secondary' are loaded into the R workspace when the library is loaded:
   
   ```r
   library(xxm)
   data(ScotsSec.xxm, package="xxm")
   ``

2. Datasets for this model are documented under *ScotsSec.xxm.RData* in ???.

3. R script for running this model is stored under *.../models/ScotsSec.1.xxm.R*
Introduction

`sleepstudy` is a two-level SEM model that includes a random intercept and a random slope. In this study, participants responded were tested for reaction time each day over the course of ten days of sleep deprivation. These responses are nested within the subject. The random intercept captures between-person differences in reaction time, where a random slope captures differences between individuals in their trajectories of change.

Creating a model: ‘sleepstudy’

The model involves two levels: `response` and `subject`

1. `response` is nested within `subject`. `response` has one observed dependent variable `rt` and one independent variable `days`.
2. `subject` is not nested within any level and has a two latent variables:
   - `intercept` is the random intercept of response `rt` at the `response` level
   - `days_slope` is the random slope of response `days` at the `response` level

`sleepstudy` is created by invoking `xxmModel()`:
```
sleepstudy <- xxmModel(levels = c("response","subject"))
```

Note: Levels are numbered according to their position in the above command. Hence, `response` corresponds to level 1 and `subject` corresponds to level 2.

Adding submodels for ‘response’ and ‘subject’

For each level declared above, we need to create a corresponding submodel. A submodel is created by invoking `xxmSubmodel()`:

1. `response`
   ```
sleep <- xxmSubmodel(model = sleep, level = "response", parents = "subject", ys = "rt", xs = "days")
   ```
2. `subject`
   ```
sleep <- xxmSubmodel(model = sleep, level = "subject", parents =, ys =, xs =, etas = c("intercept","days_slope"))
   ```

Specifying within-level model matrices for ‘response’ (level 1)

The response submodel has only one observed dependant variables(`rt`). `days` is an observed exogenous or fixed predictor. Hence it does not have any parameter matrices associated with it. Model matrices for each submodel is specified by invoking `xxmWithinMatrix()`:

1. `theta`: Observed residual-covariance matrix is a (1x1) matrix with a single element: the residual variance of `rt`.
   ```
sleep <- xxmWithinMatrix(model = sleepstudy, level = "response", type = "theta", pattern = th1_pat, value = th1_val)
   ```
Specifying within-level model matrices for ‘subject’ (level 2)

The **subject** level has one latent variable (rt). Model matrices for each submodel is specified by invoking `xxmWithinMatrix()`:

1. **psi**: Latent covariance matrix is a (2x2) matrix.
   ```r
   sleep <- xxmWithinMatrix(model = sleepstudy, level = "subject", type = "psi", pattern = ps2_pat, value = ps2_val)
   ```

2. **alpha**: Latent mean matrix is a (2x1) matrix with two elements, the average intercept of rt and the average slope of participants across days.
   ```r
   sleep <- xxmWithinMatrix(model = sleep, level = "subject", type = "alpha", pattern = al2_pat, value = al2_val)
   ```

3. Note: Within this model, the mean is not specified at the observed level using a nu matrix, but it is instead specified at the latent intercept using an ‘alpha’ matrix. Observed intercept for rt i.e., nu is fixed to 0.0.

Specifying across-level model matrices connecting 'subject' to 'response'

So far, we have specified within-level matrices. We now need to connect observed and latent variables across-levels. The loading for days from `days_slope` will be set to the value of days using the a label which includes the variable name as the label. These across-level regression connecting variables across two submodels is specified by invoking `xxmBetweenMatrix()`.

```r
subject to response
```

1. **lambda**: Factor-loading matrix is a (2x1) matrix. The loading to rt from intercept is set to 1.0 and the loading to rt from days_slope is set to the value of days.
   ```r
   sleep <- xxmBetweenMatrix(model=sleep, parent="subject", child="response", type="lambda", pattern=ly12_pat, value=ly12_val, label=ly12_label)
   ```

   Note: The second element of the label matrix is 'response.rt'. This means that the parameter is fixed to subject specific values of rt. This allows the corresponding latent variable `days_slope` to be interpreted as the random-slope of rt.

Compute: `xxmRun()`

Model specification is now complete. Parameter estimation is initiated: `sleep <- xxmRun(model = sleep)`

Notes

1. Dataset for this model is packaged with xxm in an R workspace called `sleepstudy.xxm.RData`. The two data frames called 'response' and 'subject' are loaded into the R workspace when the library is loaded:
   ```r
   library(xxm)
   data(sleepstudy.xxm, package="xxm")
   ```

2. Datasets for this model are documented under `sleepstudy.xxm.RData` in ???.

3. R script for running this model is stored under `.../models/sleepstudy.1.xxm.R`
Description

sleepstudy contains the data from the sleep deprivation study included as an example in the \texttt{lme4} package. The original data frame has been re-formatted for xxm analysis and now consists of two data frames: \texttt{sleepstudy.response} and \texttt{sleepstudy.subject}.

Usage

data(sleepstudy.response)

Format

A data frame with 180 observations on the following 4 variables.

response  a numeric vector
subject   a numeric vector
rt        a numeric vector
days      a numeric vector

See Also

\texttt{sleepstudy.subject}

Examples

data(sleepstudy.response)
## maybe \texttt{str(sleepstudy.response)} ; \texttt{plot(sleepstudy.response)} ...

Description

sleepstudy contains the data from the sleep deprivation study included as an example in the \texttt{lme4} package. The original data frame has been re-formatted for xxm analysis and now consists of two data frames: \texttt{sleepstudy.response} and \texttt{sleepstudy.subject}.

Usage

data(sleepstudy.subject)
Format

A data frame with 18 observations on the following variable.

subject a numeric vector

See Also

sleepstudy.response

Examples

data(sleepstudy.subject)
## maybe str(sleepstudy.subject) ; plot(sleepstudy.subject) ...

---

threesixty.coach  XXM Dataset: threesixty.coach

Description

threesixty involves ratings of support among hierarchically nested teachers, coaches and supercoaches. Teachers rate support that they received from coaches, and, coaches rate support that they received from supercoaches.

Usage

data(threesixty.coach)

Format

A data frame with 195 observations on the following 4 variables.

coach a numeric vector
superoach a numeric vector
coach_on_super7 a numeric vector
coach_on_super8 a numeric vector

Examples

data(threesixty.coach)
## maybe str(threesixty.coach) ; plot(threesixty.coach) ...
threesixty.supercoach  

**XXM Dataset: threesixty.supercoach**

**Description**

**threesixty** involves ratings of support among hierarchically nested teachers, coaches and supercoaches. Teachers rate support that they received from coaches, and, coaches rate support that they received from supercoaches.

**Usage**

```r
data(threesixty.supercoach)
```

**Format**

A data frame with 62 observations on the following variable.

- **supercoach** a numeric vector

**Examples**

```r
data(threesixty.supercoach)
## maybe str(threesixty.supercoach) ; plot(threesixty.supercoach) ...
```

---

threesixty.teacher  

**XXM Dataset: threesixty.teacher**

**Description**

**threesixty** involves ratings of support among hierarchically nested teachers, coaches and supercoaches. Teachers rate support that they received from coaches, and, coaches rate support that they received from supercoaches.

**Usage**

```r
data(threesixty.teacher)
```

**Format**

A data frame with 2253 observations on the following 4 variables.

- **teacher** a numeric vector
- **coach** a numeric vector
- **teach_on_coach7** a numeric vector
- **teach_on_coach8** a numeric vector
Examples

data(threesixty.teacher)
## maybe str(threesixty.teacher); plot(threesixty.teacher) ...

Introduction

threesixty is a three level SEM model about the support provided within a teacher training study. Teachers were trained by coaches, who are themselves advised by supercoaches. Each coach had multiple teachers they trained and each supercoach has multiple coaches they advised. Each teacher rated the support provided by their coach and each coach rated the support provided by their supercoach. This pattern was repeated across two years (’07 and ’08), thus there is a year1 rating and a year2 rating for each rating made. Some coaches are likely to be seen as more supportive than other coaches and so coaches will have both observed variables describing their supercoaches support and latent variables taken from the support teachers believe they give. Furthermore, some supercoaches may cause coaches to be more supportive to teachers and thus, there is an additional level of teachers ratings at the supercoach level.

Creating a model: ‘threesixty’

The model involves three levels: teacher, coach, supercoach

1. teacher is nested within coach. teacher is said to be the child level with coach as the corresponding parent level. teacher has two observed dependent variables: teach_on_coach7 and teach_on_coach8.

2. coach is nested within supercoach. coach is said to be the child level with supercoach as the corresponding parent level. coach has two observed dependent variables: coach_on_super7 and coach_on_super8.

   coach has two latent variariables: teacher7_coach and teacher8_coach.

3. supercoach is not nested within any level and is a parent of both coach. supercoach has four latent variables:
   (a) coach7_super is the random intercept of response coach_on_super7 at the coach level.
   (b) coach8_super is the random intercept of response coach_on_super8 at the coach level .
   (c) teach7_super is the random intercept of latent variable teacher7_coach at the coach level.
   (d) teach8_super is the random intercept of latent variable teacher8_coach at the coach level.

threesixty is created by invoking xxmModel().

threesixty <- xxmModel(levels = c("teacher","coach","supercoach"))

Note: Levels are numbered according to their position in the above command. Hence, teacher corresponds to level 1, coach corresponds to level 2, and supercoach corresponds to level 3. Lower levels are nested beneath higher levels.
Adding submodels for ‘teacher‘, ‘coach‘ and ‘supercoach‘

For each level declared above, we need to create corresponding submodels. Each submodel contains all the elements that will exist within a level. A submodel is created by invoking xxmSubmodel():

1. teacher
   threesixty <- xxmSubmodel(model = threesixty, level = "teacher", parents = "supercoach", ys = c("teach_on_coach7","teach_on_coach8"), xs = , etas = , data = threesixty.teacher)

2. coach
   threesixty <- xxmSubmodel(model = threesixty, level = "coach", parents = "supercoach", ys = c("coach_on_super7","coach_on_super8"), xs = , etas = c("teacher7_coach","teacher8_coach"), data = threesixty.coach)

3. supercoach
   threesixty <- xxmSubmodel(model = threesixty, level = "supercoach", parents = , ys = , xs = , etas = c("coach7_super","coach8_super","teacher7_super","teacher8_super"), data = threesixty.supercoach)

Specifying within-level model matrices for ‘teacher‘ (level 1)

The teacher submodel has two observed variables (teach_on_coach7 and teach_on_coach8). Model matrices for each submodel is specified by invoking xxmWithinMatrix():

1. theta: Observed residual-covariance matrix is a (2x2) matrix with a three unique elements, the residual variance of teach_on_coach7, the residual variance of teach_on_coach8, and the residual covariance between teach_on_coach7 and teach_on_coach8.
   threesixty <- xxmWithinMatrix(model = threesixty, level = "teacher", type = "theta", pattern = th1_pat, value = th1_val)

2. nu: Observed variable intercept matrix is a (2x1) matrix with two elements, the intercept of teach_on_coach7 and the intercept of teach_on_coach8.
   threesixty <- xxmWithinMatrix(model = threesixty, level = "teacher", type = "nu", pattern = nu1_pat, value = nu1_val)

Specifying within-level model matrices for ‘coach‘ (level 2)

The most complex of the levels in the current model is the coach level because it has both observed and latent variables. Thus, we will need to include within-level matrices for both latent (psi) and observed (theta and nu) variables. Model also include a lambda matrix. Model matrices for each submodel is specified by invoking xxmWithinMatrix():

1. theta: Observed residual-covariance matrix is a (2x2) matrix with a three unique elements, the residual variance of coach_on_super7, the residual variance of coach_on_super8, and the residual covariance between written and course.
   threesixty <- xxmWithinMatrix(model = threesixty, level = "coach", type = "theta", pattern = th2_pat, value = th2_val)

2. nu: Observed variable intercept matrix is a (2x1) matrix with two elements, the intercept of coach_on_super7 and the intercept of coach_on_super8.
   threesixty <- xxmWithinMatrix(model = threesixty, level = "coach", type = "nu", pattern = nu2_pat, value = nu2_val)

3. psi: Latent covariance matrix is a (2x2) matrix. The elements include variances of the two latent variables (teacher7_coach and teacher8_coach) and the covariances between them.
   threesixty <- xxmWithinMatrix(model = threesixty, level = "coach", type = "psi", pattern = ps2_pat, value = ps2_val)

4. lambda: Latent covariance matrix is a (2x2) matrix. The columns represent the two latent variables (teacher7_coach and teacher8_coach) and the rows represent the two observed variables (coach_onSuper7 and coach_onSuper8). All elements are estimated.
   threesixty <- xxmWithinMatrix(model = threesixty, level = "coach", type = "lambda", pat
Specifying within-level model matrices for ‘supercoach’ (level 3)

`supercoach` has four latent variables `teacher7_super`, `teacher8_super`, `coach7_super`, and `coach8_super`. This level includes a single latent covariance matrix.

1. `psi`: Latent covariance matrix is a (4x4) matrix. The elements include variances of the four latent variables and the six unique covariances between them.

```r
threesixty <- xxmWithinMatrix(model = threesixty, level = "supercoach", type = "psi", pattern = ps3_pat, value = ps3_val)
```

Specifying across-level model matrices connecting 'teacher' and 'coach' and 'supercoach'

So far, we have specified within-level matrices. We now need to connect observed and latent variables across-levels. Two types of across-level matrices exist. Latent variables with observed variables as indicators will use a lambda matrix. Latent variables with other latent variables as indicators will use a beta matrix. These across-level regression connecting variables across two submodels is specified by invoking `xxmBetweenMatrix()`.

Note: The direction of influence is from the higher level model `supercoach` called the **parent** to the lower level model `teacher` or `coach` called the **child**. Furthermore, in each between matrix, all the elements of the connection type must be included within the matrix. The lambda and beta matrices connected observed variables to latent variables and thus the matrix must then be 2x4. 2 for the two indicators (either observed or latent) at `coach` and 4 for the four latent variables at `supercoach`.

Specifying across-level model matrices connecting 'supercoach' to 'teacher'

1. `lambda`: Factor-loading matrix is a (2x4) matrix. It has eight different elements, six of which are set to 0 and two of which are set to 1.0.

```r
threesixty <- xxmBetweenMatrix(model = threesixty, parent = "supercoach", child = "teacher", type = "lambda", pattern = l23_pat, value = l23_val)
```

Specifying across-level model matrices connecting 'supercoach' to 'coach'

1. `beta`: Factor-loading matrix is a (2x4) matrix. It has eight different elements, six of which are set to 0 and two of which are set to 1.0.

```r
threesixty <- xxmBetweenMatrix(model = threesixty, parent = "supercoach", child = "coach", type = "beta", pattern = be23_pat, value = be23_val)
```

Compute: `xxmRun()`

Model specification is now complete. Parameter estimation is initiated: `threesixty <- xxmRun(model = threesixty)`

Notes

1. Dataset for this model is packaged with `xxm` in an R workspace called `threesixty.xxm.RData`. The three data frames called ‘teacher’, ‘coach’, and ‘supercoach’ are loaded into the R workspace when the library is loaded:

```r
library(xxm)
data(threesixty.xxm, package="xxm")
```

2. Datasets for this model are documented under `threesixty.xxm.RData` in ???.

3. R script for running this model is stored under `.../models/threesixty.0.xxm.R`
Description

1. xxmBetweenMatrix is used to specify regression relationships among observed and/or latent variables of two levels related by a parent-child relationship.

There are four types of regression relationships that can be defined across levels.

(a) Observed on latent variables or measurement relationship: Observed dependent variables at a child level are regressed on latent variable at a parent level. In SEM, such relationships are commonly referred to as measurement relationships. The matrix representing observed on latent regression is called "Lambda".

(b) Observed on exogenous observed predictors: Observed dependent variables at a child level are regressed on exogenous observed predictors at a parent level. The matrix representing observed on observed regression is called "Kappa".

(c) Latent variable regression: Latent dependent variables at a child level are regressed on latent predictors at a parent level. The matrix representing observed on observed regression is called "Beta".

(d) Latent on exogenous observed predictors: Latent dependent variables at a child level are regressed on exogenous observed predictors at a parent level. The matrix representing observed on observed regression is called "Gamma".

2. xxmBetweenMatrix is also used to specify covariance relationships among observed and/or latent variables across levels related by a ‘sibling’ relationship.

(a) Covariance among observed dependent variables: The matrix representing observed residual-covariance is called "Theta".

(b) Covariance among latent dependent variables: The matrix representing observed residual-covariance is called "Psi".

Usage

xxmBetweenMatrix(model, parent, child, type, pattern, value, label, name)

Arguments

model Name of the xxmModel.

parent Name of the parent level. Parent level is the one with the predictors. Predictors may be latent (etas) or observed (xs).

child Name of the child level. Child level is the one with the dependent variables. Dependent or outcome variables may be observed (ys) or latent (xs).

type Type of the matrix specified.

1. Valid types for regression include "lambda", "beta", "kappa", "gamma".
2. Valid types for covariance include "psi", "theta".
pattern

Name of an integer matrix used to declare free and fixed parameters. A pattern matrix is an R integer matrix containing 1s (free) or 0s (fixed). Elements in the pattern matrix that are "1" are freely estimated. Elements in the pattern matrix that are "0" are correspondingly fixed at the values provided in the value matrix.

value

Name of a real matrix used to declare start values for freely estimated parameters and constant values for fixed parameters.

label

Name of a character matrix used to label individual parameters. Labels are used for imposing equality constraints. Two parameters with the same label will be constrained to be equal. Label matrix is optional, if no label matrix is provided, xwm will generate informative labels for all parameters.

name

Name of the xwmBetweenMatrix. Name is optional. If no name is provided, xwm will generate an informative name.

Details

xwmBetweenMatrix is used to specify regression (measurement and structural) relationships among observed and/or latent variables across two levels related by a parent-child relationship. Obviously, a parent-child relationship between the two levels must be previously declared using the parent argument of the xwmSubmodel command when declaring the child level.

xwmBetweenMatrix can also be used to specify covariance relationship among observed and/or latent variables across two levels related by a sibling relationship.

Value

xwmBetweenMatrix returns the model object that was passed to it as its first argument.

Note

1. Parent-child relationship must exist between the two levels used in this command.
2. Pattern matrix must be present in the R workspace before being used in this command.
3. At present, the xwmBetweenMatrix function is limited to directional type (i.e., type = "lambda"; type = "beta", type = "gamma"; type = "kappa").

See Also

xwmModel, xwmSubmodel, xwmWithinMatrix.

Examples

## Not run:
ly12_pat <- matrix(c(0,1,1,0,0,0,0,0,1,1),6,2)
ly12_val <- matrix(c(1.1,.9,.8,/zero.noslash,/zero.noslash,/zero.noslash,/zero.noslash,/zero.noslash;/zero.noslash,1,1.2,1.3),6,2)
mymodel <- xwmBetweenMatrix(model = mymodel, parent = "school", child = "student", type = "lambda", pattern = ly12_pat, value = ly12_val, label = , name = )

## End(Not run)
Describes a new xxm Model.

Usage

```
xxmModel(levels=)
```

Arguments

- **levels**: Is a vector of names for the levels to be included in the model. Names must be listed in ascending hierarchical order (bottom-up) for nested models. For examples, in a three level model with students nested within teachers, and classrooms nested within schools, levels are ("student", "teacher", "school").

Details

The `xxmModel` is the first step necessary for specifying a multi-level SEM model. Each of the levels listed in the `levels` argument must appear in a subsequent `xxmSubmodel` command.

Value

Returns a handle to the model created by the command. Technically, it is an R object of type `externalPointer`. This handle can be named (e.g., "l2RanSlp" or "dog"). Whatever name is used for the return value is passed to all subsequent model commands as the value of the `model` parameter.

Note

`xxmModel` simply defines all the levels present in the model. For each of the level listed in `xxmModel`, a proper submodel must be specified using `xxmSubmodel` command.

Examples

```
## Not run:
library(xxm)
myModel <- xxmModel(levels=c("student","school"))
ccModel<- xxmModel(levels=c("response","person","scale"))
l4xxm <- xxmModel(levels=c("student","classroom","school","neighborhood"))
## End(Not run)
```
A birds-eye view of model construction in xxm.

Details

xxm uses LISREL like matrices for specifying parameters of multilevel-SEM matrices. A typical single level SEM model requires specification of (a) a measurement model, (b) various regression or structural relationships among observed and latent variables, (c) covariances or residual covariances among observed and/or latent variables, and (d) means/intercepts for the observed and/or latent variables.

With multiple levels, we now have to consider one such model for each level. Furthermore, we can also think of regression relationships across any two levels related by a parent-child relationship. Obviously, with so many possibilities the model specification can get confusing. In order to reduce the burden of model specification, xxm takes a structured approach in which a complete model is put-together or constructed in a LEGO like fashion using simpler objects. Each object itself can be constructed using the same set of simple and obvious rules.

Three classes of xxm objects

There are three broad classes of xxm objects:

1. Model: xxmModel is just a 'container' for holding submodels for each level and related matrices. There is a single xxmModel object created using xxmModel command. Details of creating main model object are provided in xxmModel.

2. Submodel: xxmSubmodel is also a container. There is one xxmSubmodel corresponding to each level. Each xxmSubmodel is created by invoking a xxmSubmodel command. Obviously, a level specific xxmSubmodel holds parameter matrices for that level. Details of creating a submodel object are provided in xxmSubmodel.

3. Parameter matrices: Parameters to be estimated are specified using model matrices. There are two types of matrices:
   (a) Within-level matrices specify a typical SEM model for each level and are constructed by xxmWithinMatrix.
   (b) Between-level matrices specify an extended ML-SEM model and connect two different levels and are constructed by invoking xxmBetweenMatrix.

Other than this simple within- vs. between-matrix distinction, the matrix itself is constructed similarly. Model matrices are described at length in xxmModelMatrices. The commands for constructing model matrices are provided in xxmWithinMatrix and xxmBetweenMatrix.

These simple commands are adequate for specifying a complex model with many levels.

See Also

level, xxmModelMatrices, xxmModel, xxmSubmodel, xxmWithinMatrix, xxmBetweenMatrix.
Description

Description of xxm matrices.

Details

xxm uses LISREL-like matrices for specifying parameters of multilevel-SEM matrices. A typical single level SEM model requires specification of (a) a measurement model, (b) various regression or structural relationships among observed and latent variables, (c) covariances or residual covariances among observed and/or latent variables, and (d) means/intercepts for the observed and/or latent variables.

1. What are within- and between-matrices?

With multiple levels, we have a submodel for each level. A within-level matrix is used for specifying submodel at each level.

With ML-SEM, we can also specify relationships among variables across two different levels. A between-matrix is used for specifying linkages across two different levels.

2. What types of parameter matrices are available in xxm? I. Within-level matrices:

(A) Regression relationships among observed and/or latent variables within a given level.

There are four types of regression relationships that can be defined within each level.

(1) Observed on latent variables or measurement relationship: Observed dependent variables are regressed on latent variable. In SEM, such relationships are referred to as measurement relationships. The matrix representing observed on latent regression is called "Lambda".

(2) Observed on exogenous observed predictors: Observed dependent variables are regressed on exogenous observed predictors. The matrix representing observed on observed regression is called "Kappa".

(3) Latent variable regression: Latent dependent variables are regressed on latent predictors. The matrix representing observed on observed regression is called "Beta".

(4) Latent on exogenous observed predictors: Latent dependent variables are regressed on exogenous observed predictors. The matrix representing observed on observed regression is called "Gamma".

(B) Covariances/residual-covariances among observed and latent variables

(1) Covariance among observed dependent variables: The matrix representing observed residual-covariance is called "Theta".

(2) Covariance among latent dependent variables: The matrix representing observed residual-covariance is called "Psi".

(C) Means/Intercepts of observed and latent variables

(1) Intercepts of observed dependent variables: The matrix representing observed residual-covariance is called "Nu".
(2) Means/Intercepts of latent variables: The matrix representing observed residual-covariance is called "Alpha".

II. Between-level matrices:

(A) xxmBetweenMatrix is used to specify regression relationships among observed and/or latent variables of two levels related by a parent-child relationship.

There are four types of regression relationships that can be defined across levels.

(1) Observed on latent variables or measurement relationship: Observed dependent variables at a child level are regressed on latent variable at a parent level. In SEM, such relationships are commonly referred to as measurement relationships. The matrix representing observed on latent regression is called "Lambda".

(2) Observed on exogenous observed predictors: Observed dependent variables at a child level are regressed on exogenous observed predictors at a parent level. The matrix representing observed on observed regression is called "Kappa".

(3) Latent variable regression: Latent dependent variables at a child level are regressed on latent predictors at a parent level. The matrix representing observed on observed regression is called "Beta".

(4) Latent on exogenous observed predictors: Latent dependent variables at a child level are regressed on exogenous observed predictors at a parent level. The matrix representing observed on observed regression is called "Gamma".

(B) xxmBetweenMatrix is also used to specify covariance relationships among observed and/or latent variables across levels related by a 'sibling' relationship.

(1) Covariance among observed dependent variables: The matrix representing observed residual-covariance is called "Theta".

(2) Covariance among latent dependent variables: The matrix representing observed residual-covariance is called "Psi".

3. What are the parts of a parameter matrix?

The previous section described various types of parameter matrices. Regardless of the type of the matrix, parameter matrices are constructed similarly. Each parameter matrix has three components: (a) pattern matrix, (b) value matrix, and (c) label matrix. These matrices make sense as a whole. In the following description, keep in mind that the pattern matrix is meaningful only if the value matrix is available. You will have to read the following section twice before it will all make sense.

(1) Pattern matrix: A pattern matrix is used to indicate if a parameter is to be estimated or if it is to be fixed to a known value. For example, typically in a factor-loading matrix the first element is fixed to 1.0 for defining the latent variable scale, where as, the remaining elements are freely estimated. Such information is provided using a pattern matrix. The pattern matrix contains either "0" or "1". If the element of a pattern matrix is "0", then the corresponding element is not estimated or fixed. If the element of a pattern matrix is "1", then the corresponding element is estimated. For the previous example, if the first element is to be fixed to a value "1.0", the first element of the pattern matrix would be "0". The pattern matrix itself only provides information about which parameter to fix or estimate. However, if the parameter is to be "fixed", we still need to provide the actual value at which to fix the parameter separately. This is accomplished by the value matrix.

(2) Value matrix: A value matrix is used to provide constant values to be used for fixed parameters and plausible ‘start’ values for the free parameters. For the above example, the first element of the
value matrix would be "1.0". Taken together, pattern matrix tells \texttt{xxm} to "fix" the first element, and the value matrix tells \texttt{xxm} to fix the value to "1.0".

(3) Label matrix: A label matrix is used to provide informative labels to each parameter. There are no special rules for the choice of parameter labels and the matrix itself is optional. If no label matrix is provided, \texttt{xxm} will construct one with suitably informative labels.

Label matrices are used primarily for imposing equality constraints. For example, in investigating measurement invariance across groups, we may wish to constrain factor-loadings to be equal across the two groups. This is accomplished by using the same parameter label for the corresponding elements of the factor-loading matrix.

4. How are matrices constructed in R? The three matrices described above are constructed as regular R matrices.

(a) A within-level factor-loading matrix for a level ('student') with four observed variables and a single latent variable may be constructed as follows:

\begin{verbatim}
lambda_pattern <- matrix(c(0,1,1,1), 4, 1) #First element of the pattern matrix is "fixed", the remaining three elements are "free" lambda_value <- matrix(c(1.0,.9,1.1,1.2), 4, 1) #First element of the value matrix is "1.0", meaning that the first element is to fixed at value 1.0. #The remaining three values (.9, 1.1, 1.2) are called 'start' values. These represent our best initial guess regarding these factor-loadings. \texttt{xxm} will use these values to begin the iterative estimation process to estimate the final values. lambda_label<- matrix (c("l_11","l_21", "dog", "cat"), 4, 1)
\end{verbatim}

Once the three component matrices are constructed the parameter matrix is created by invoking \texttt{xxmWithinMatrix} command:

\begin{verbatim}
myModel <- xxmWithinMatrix(model = myModel, level = "student", type = "lambda", pattern = lambda_pattern, value = lambda_value, label = lambda_lable, name = "StudentFactor-loading")
\end{verbatim}

The above statement can be read as: Add a matrix of type 'lambda' to 'student' submodel contained within 'myModel' using 'lambda_pattern', 'lambda_value', and 'lambda_label' as the three components and call it 'StudentFactor-loading'.

(b) A similar between-level factor-loading matrix connecting child-level ('student') with parent-level ('teacher') may be constructed as follows:

\begin{verbatim}
lambda_pattern <- matrix(c(0,1,1,1), 4, 1) #First element of the pattern matrix is "fixed", the remaining three elements are "free" lambda_value <- matrix(c(1.0,.9,1.1,1.2), 4, 1) #First element of the value matrix is "1.0", meaning that the first element is to fixed at value 1.0. #The remaining three values (.9, 1.1, 1.2) are called 'start' values. These represent our best initial guess regarding these factor-loadings. \texttt{xxm} will use these values to begin the iterative estimation process to estimate the final values. lambda_label<- matrix (c("s_t_l11","s_t_l21", "s_t_l31", "s_t_l41"), 4, 1)
\end{verbatim}

Once the three component matrices are constructed the parameter matrix is created by invoking \texttt{xxmBetweenMatrix} command:

\begin{verbatim}
myModel <- xxmWithinMatrix(model = myModel, parent= "teacher", child = "student", type = "lambda", pattern = lambda_pattern, value = lambda_value, label = lambda_lable, name = "StudentTeacherFactor-loading")
\end{verbatim}

The above statement can be read as: Add a matrix of type 'lambda' connecting observed variables in child submodel for 'student' with latent variables in parent submodel for 'teacher' contained within 'myModel' using 'lambda_pattern', 'lambda_value', and 'lambda_label' as the three components and call it 'StudentTeacherFactor-loading'.
See Also

`xxmModel` `xxmSubmodel`

xxmRun

Estimate a Given xxm Model

Description

xxmRun is used to estimate a specified xxmModel

Usage

xxmRun(model)

Arguments

model Name of the xxmModel.

Details

Once the model is constructed using `xxmModel`, `xxmSubmodel`, `xxmWithinMatrix`, and `xxmBetweenMatrix`, model parameters are estimated by invoking xxmRun.

Value

As always, the function returns the model object.

Note

A complete model must be specified before invoking xxmRun. In particular, it is useful to invoke one command at a time to identify potential issues in model specification.

See Also

`xxmModel` `xxmSubmodel` `xxmWithinMatrix` `xxmBetweenMatrix`

Examples

```r
## Not run: xxmRun(model = mymodel)
```
_create_a_new_xxm_submodel

Description

Defines the parameters and variables of a new xxm Submodel.

Usage

xxmSubmodel(model, level, parents, ys, xs, etas, data, siblings)

Arguments

- **model**: Name of the xxmModel to which this Submodel is to be added.
- **level**: Name of the level defined in this Submodel. The level must be previously declared in a xxmModel statement.
- **parents**: A character vector of Name(s)/IDs of level(s) at a higher level than the current submodel. A level is identified as a parent if that level includes predictors of variables in the current level. All levels listed as a parent to the current level must have corresponding parent-ID columns in the dataset (See below).
- **ys**: A character vector of names of all observed dependent variables at this level. Dataset for the current level must include columns for these variables.
- **xs**: A character vector of names of all observed exogenous predictor variables at this level. Dataset for the current level must include columns for these variables.
- **etas**: A character vector of names of all latent variables to be included in the submodel. Latent variables in the list must follow a specific order. Latent dependent variables must be listed before latent independent variables. A latent variable can only influence those that latent variables that precede the variable in the list. For example, with etas = c("eta1", "eta2"), "eta1" may be regressed on "eta2", but "eta2" may not be regressed on "eta1".
- **data**: A dataframe containing observed variables for the current level. The dataframe must include following columns: (1) ID column for the current level. Column name of the ID variable column must match the name of the current level. This is the name listed in the xxmModel statement and the value supplied to the level argument of the xxmSubmodel statement. (2) One or more Parent-ID columns. There must be as many columns with IDs of parents as the number of parents listed in the parents argument. Column names of the parent-ID columns must match the names of the corresponding levels. (3) Observed dependent variables. The dataset must include columns for all the variables listed in ys. (4) Observed exogenous predictors. The dataset must include columns for all the variables listed in xs.
- **siblings**: A vector of names of one or more other levels that are related to the current level in a one-to-one sibling relationship.
Details

The submodel command declares variables (observed and latent) and data for a level. For each level listed in xxmModel, a corresponding submodel must be created using the submodel command.

Value

Submodel returns the model object that was passed to it as its first argument. In fact, all commands in xxm pass name of the model as its first argument.

Note

Invoking submodel provides xxm with the information necessary for providing informative feedback in case of user error.

Examples

```r
## Not run:
library(xxm)
data(brim)
xm <- xxmModel(levels = c("school", "student"))
xm <- xxmSubmodel(model = xm,
                   level = "student",
                   parents = c("school"),
                   ys = c("y1","y2"),
                   xs = ,
                   etas = ,
                   data = student,
                   siblings = )

xm <- xxmSubmodel(model = xm,
                   level = "school",
                   parents = ,
                   ys = ,
                   xs = ,
                   etas = c("eta1","eta2"),
                   data = school,
                   siblings = )

## End(Not run)
```

### xxmWithinMatrix

**Specifying Submodel for a Level**

**Description**

xxmWithinMatrix is used to specify model parameters of a typical SEM model within each level present in the model.
1. Regression relationships among observed and/or latent variables within a given level. There are four types of regression relationships that can be defined within each level.
   (a) Observed on latent variables or measurement relationship: Observed dependent variables are regressed on latent variable. In SEM, such relationships are referred to as measurement relationships. The matrix representing observed on latent regression is called "Lambda".
   (b) Observed on exogenous observed predictors: Observed dependent variables are regressed on exogenous observed predictors. The matrix representing observed on observed regression is called "Kappa".
   (c) Latent variable regression: Latent dependent variables are regressed on latent predictors. The matrix representing observed on observed regression is called "Beta".
   (d) Latent on exogenous observed predictors: Latent dependent variables are regressed on exogenous observed predictors. The matrix representing observed on observed regression is called "Gamma".

2. Covariances/residual-covariances among observed and latent variables
   (a) Covariance among observed dependent variables: The matrix representing observed residual-covariance is called "Theta".
   (b) Covariance among latent dependent variables: The matrix representing observed residual-covariance is called "Psi".

3. Means/Intercepts of observed and latent variables
   (a) Intercepts of observed dependent variables: The matrix representing observed residual-covariance is called "Nu".
   (b) Means/Intercepts of latent variables: The matrix representing observed residual-covariance is called "Alpha".

Usage

`xxmWithinMatrix(model, level, type, pattern, value, label, name)`

Arguments

- **model**: Name of the `xxmModel`.
- **level**: Name of the current level for which the submodel is being defined. The level must be declared in a previous `xxmModel` statement.
- **type**: Type of the matrix specified.
  1. Valid types for regression matrices include "lambda", "beta", "kappa", and "gamma".
  2. Valid types for covariance matrices include "psi", and "theta".
  3. Valid types for mean/intercept matrices include "alpha", and "nu".
- **pattern**: Name of an integer matrix used to declare free and fixed parameters. A pattern matrix is an R integer matrix containing 1s (free) or 0s (fixed). Elements in the pattern matrix that are "1" are freely estimated. Elements in the pattern matrix that are "0" are correspondingly fixed at the values provided in the value matrix.
- **value**: Name of a real matrix used to declare start values for freely estimated parameters and constant values for fixed parameters.
label  Name of a character matrix used to label individual parameters. Labels are used for imposing equality constraints. Two parameters with the same label will be constrained to be equal. Label matrix is optional, if no label matrix is provided, \texttt{xxm} will generate informative labels for all parameters.

name   Name of the \texttt{xxmWithinMatrix}. Name is optional. If no name is provided, \texttt{xxm} will generate an informative name.

\textbf{Value}

\texttt{xxmWithinMatrix} returns the model object that was passed to it as its first argument.

\textbf{Note}

1. Level must exists before the command can be invoked. This means that the level is declared in \texttt{xxmModel} and defined by invoking \texttt{xxmSubmodel}.

2. All levels must be defined using \texttt{xxmSubmodel} command before any matrices are added to any of the models.

3. Dimensions of matrices must appropriately match the number of observed dependent, observed exogenous predictors and latent variables declared for the level. For example, in a level with 6 observed variables and 2 latent variables, a "lambda" matrix must have 6 rows and 2 columns.

\textbf{See Also}

\texttt{xxmModel, xxmSubmodel, xxmBetweenMatrix}.

\textbf{Examples}

\begin{verbatim}
## Not run:
mymodel <- xxmWithinMatrix(model = mymodel, level = "student", type = "nu", pattern = nu1_pat, value = nu1_val, label = nu1_label, name = nu1)
## End(Not run)
\end{verbatim}
Index

*Topic I/O Psychology
  rolemodels.1.xxm, 45

*Topic command
  xxmBetweenMatrix, 63
  xxmModel, 65
  xxmRun, 70
  xxmSubmodel, 71
  xxmWithinMatrix, 72

*Topic complex dependency
  rolemodels.1.xxm, 45

*Topic concepts
  level, 34
  xxmModelConstruction, 66
  xxmModelMatrices, 67

*Topic cross-classified
  faces.xxm.1, 7
  ScotsSec.xxm.1, 52

*Topic datasets
  brim.school, 3
  brim.student, 3
  ex98.l1, 4
  ex98.l2, 5
  faces.rater, 5
  faces.response, 6
  faces.target, 7
  gcsemv.boy, 11
  gcsemv.girl, 12
  gcsemv.school, 12
  hcfa.school, 20
  hcfa.student, 20
  hcfa.teacher, 21
  ltranslp.l1, 35
  ltranslp.l2, 35
  lwid.first, 36
  lwid.kinder, 37
  lwid.response, 37
  lwid.school, 38
  lwid.second, 39
  lwid.student, 39
  mlcfa.student, 40
  mlcfa.teacher, 40
  pcwa.student, 41
  pcwa.teacher, 42
  rolemodels.individual, 48
  rolemodels.response, 49
  rolemodels.team, 50
  ScotsSec.primary, 50
  ScotsSec.secondary, 51
  ScotsSec.student, 52
  sleepstudy.response, 57
  sleepstudy.subject, 57
  three sixty.coach, 58
  three sixty.supercoach, 59
  three sixty.teacher, 59

*Topic dyadic
  three sixty.xxm.1, 60

*Topic education
  ScotsSec.xxm.1, 52
  three sixty.xxm.1, 60

*Topic equality-constraint
  hcfa.xxm.2, 26

*Topic growth curves
  sleepstudy.1.xxm, 55

*Topic latent variable
  random-intercept
  hcfa.xxm.3, 30

*Topic model
  faces.xxm.1, 7
  gcsemv.xxm.1, 13
  gcsemv.xxm.2, 17
  hcfa.xxm.1, 22
  hcfa.xxm.2, 26
  hcfa.xxm.3, 30
  pcwa.xxm.1, 42
  rolemodels.1.xxm, 45
  ScotsSec.xxm.1, 52
  sleepstudy.1.xxm, 55
  three sixty.xxm.1, 60
*Topic multiple-groups
gcemv.xxm.1, 13
gcemv.xxm.2, 17

*Topic random-intercept
cpwa.xxm.1, 42

*Topic random-slopes
sleepstudy.1.xxm, 55

*Topic repeated-measures
threesixty.xxm.1, 60

*Topic three-level
gcemv.xxm.1, 13
gcemv.xxm.2, 17
hcfa.xxm.1, 22
hcfa.xxm.2, 26
hcfa.xxm.3, 30
threesixty.xxm.1, 60

*Topic two-level
cpwa.xxm.1, 42

brim.school, 3, 4
brim.student, 3, 3

Cross-Classified (ScotsSec.xxm.1), 52

ex98.l1, 4, 5
ex98.l2, 4, 5

faces.rater, 5, 6, 7
faces.response, 6, 6, 7
faces.target, 6, 7
faces.xxm.1, 7

gcemv.boy, 11, 12, 13
gcemv.girl, 11, 12, 13
gcemv.school, 11, 12, 12
gcemv.xxm.1, 13
gcemv.xxm.2, 16, 17

hcfa (hcfa.school), 20
hcfa.school, 20, 21
hcfa.student, 20, 20, 21
hcfa.teacher, 20, 21, 21
hcfa.xxm.1, 22, 29, 33
hcfa.xxm.2, 25, 26, 33
hcfa.xxm.3, 25, 29, 30

level, 34, 66
ltranslp.l1, 35, 36
ltranslp.l2, 35, 35
lwid.first, 36

llwid.kinder, 37
lwid.response, 37
lwid.school, 38
lwid.second, 39
lwid.student, 39

mlcfa.student, 40, 41
mlcfa.teacher, 40, 40
pcwa.student, 41, 42
pcwa.teacher, 41, 42
pcwa.xxm.1, 42
rolemodels.1.xxm, 45
rolemodels.individual, 48
rolemodels.response, 49
rolemodels.team, 50
ScotsSec.primary, 50, 51, 52
ScotsSec.secondary, 51, 51, 52
ScotsSec.student, 51, 52
ScotsSec.xxm.1, 52
sleepstudy.1.xxm, 55
sleepstudy.response, 57, 58
sleepstudy.subject, 57, 57
threesixty.coach, 58
threesixty.suppercoach, 59
threesixty.teacher, 59
threesixty.xxm.1, 60

xxmBetweenMatrix, 63, 66, 70, 74
xxmModel, 34, 64, 65, 66, 70, 74
xxmModelConstruction, 66
xxmModelMatrices, 66, 67
xxmRun, 70
xxmSubmodel, 34, 64, 66, 70, 71, 74
xxmWithinMatrix, 64, 66, 70, 72