xxm Reference Manual

December 26, 2013

Type Package

Title Structural Equation Modeling for Dependent Data

Version 0.6.0

Date 2013-12-01

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.student` *XXM Dataset: brim.student*

Description

brim contains two data frames: **brim.student** and **brim.teacher**, 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
`teacher`	a numeric vector
`y1`	a numeric vector
`y2`	a numeric vector

See Also

`brim.teacher`

Examples

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

`brim.teacher` *XXM Dataset: brim.teacher*

Description

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

Usage

```
data(brim.teacher)
```

Format

A data frame with 25 observations on the following variable.

`teacher` a numeric vector

See Also

`brim.student`

Examples

```
data(brim.teacher)
## maybe str(brim.teacher) ; plot(brim.teacher) ...
```

`ex98.l1` *XXM Dataset: ex98.l1*

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.l1** and **ex98.l2**.

Usage

```
data(ex98.l1)
```

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.l2`

Examples

```
data(ex98.l1)
## maybe str(ex98.l1) ; plot(ex98.l1) ...
```

`ex98.l2` *XXM Dataset: ex98.l2*

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.l1** and **ex98.l2**.

Usage

```
data(ex98.l2)
```

Format

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

`l2` a numeric vector

`w` a numeric vector

See Also

`ex98.l1`

Examples

```
data(ex98.l2)
## maybe str(ex98.l2) ; plot(ex98.l2) ...
```

`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

```
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) ...
```

`faces.target` *XXM Dataset: faces.target*

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) ...
```

`faces.xxm.1` *XXM Example Model: faces.xxm.1.R*

Introduction

faces is a three level SEM model for rater-target research designs: participants (raters) respond to a series of randomly sampled stiumli (targets). Specifically, responses are simultaniously 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 photograps 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 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 targets that score 'high' on target_SYM are rated as more symmetrical than average by all raters.

   faces is created by invoking `xxmModel()`.

   ```
   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**

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

2. **rater**

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

3. **target**

   ```
   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,.0,.0,2),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 inetrcepts 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)`

```
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:

`faces <-xxmRun(model = faces)`

Note

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

`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

```
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

```
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

```
data(hcfa.student)
```

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) ...
```

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

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**

   ```
   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 frely estimated.

   ```
   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 **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.

```
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(mdoel = 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 rergession 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.

   ```
   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 lLatent 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.

   ```
    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:

```
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:
   `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`.

`hcfa.xxm.2` *xxM Model Example: hcfa.xxm.2.R*

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 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**

```
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 frely 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.

```
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(mdoel = 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 rergession 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.

   ```
   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:

   `ly1_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 lLatent 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.

   ```
    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:

   `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:

```
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:
   `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`.

`hcfa.xxm.3` *xxM Model Example: hcfa.xxm.3.R*

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: **y1**, **y2**, and **y3**; and a single latent variable **A1**.
2. **teacher** is a **parent** of **student** and is itself nested within **school**. **teacher** has no observed variable and a single latent variable: **A2** defined by **A1** at the **student** level.
3. **school** is a **parent** of **teacher**. **school** includes three observed dependent variables: **q1**, **q2**, and **q3**. **school** submodel has two latent variables: **A3** defined by latent variable **A2** at the **teacher** 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"), ys = ys1, xs =,
    etas = c("A1"), data = hcfa.student)
   ```

2. **teacher**

   ```
   teacher <- xxmSubmodel(model = hcfa, level="teacher",
     parents = c("school"), 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 frely estimated.

   ```
   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 **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.

```
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(mdoel = 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 rergession 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 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**.

   ```
   hcfa <- xxmBetweenMatrix(model = hcfa, parent = "teacher", child = "student",
   type = "beta", pattern = be12_pat, value = be12_val)
   ```

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 The second latent variable **Q** was already defined by level-3 observed variables. Hence the second element is fixed to 0.0.

   ```
    hcfa <- xxmBetweenMatrix(model = hcfa, parent = "school",
     child = "teacher", type = "beta", pattern = be23_pat,
     value = be23_val)
   ```

Compute: xxmRun()

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

```
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:

   `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.3.xxm.R`

See Also

`hcfa.xxm.1`, `hcfa.xxm.2`.

`level` *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 numtiple exchangable 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 `tecaher` 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 exmaple, 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 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`

`lranslp.l1` *XXM Dataset: lranslp.l1*

Description

lranslp contains two data frames: **lranslp.l1** and **lranslp.l2**, that have been formatted for xxM analysis. These data were 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(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

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

`lranslp.l2` *XXM Dataset: lranslp.l2*

Description

lranslp contains two data frames: **lranslp.l1** and **lranslp.l2**, that have been formatted for xxM analysis. These data were 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(lranslp.l2)
```

Format

A data frame with 150 observations on the following variable.

`l12` a numeric vector

See Also

`lranslp.l1`

Examples

```
data(lranslp.l2)
## maybe str(lranslp.l2) ; plot(lranslp.l2) ...
```

`lwid.first` *XXM Dataset: 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.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) ...
```

`lwid.kinder` *XXM Dataset: 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.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) ...
```

`lwid.response` *XXM Dataset: lwid.response*

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` *XXM Dataset: 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

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

Format

A data frame with 150 observations on the following variable.

`second` a numeric vector

Examples

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

`lwid.student` *XXM Dataset: 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

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

Format

A data frame with 1858 observations on the following variable.

`student` a numeric vector

Examples

```
data(lwid.student)
## maybe str(lwid.student) ; plot(lwid.student) ...
```

`mlcfa.student` *XXM Dataset: mlcfa.student*

Description

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

Usage

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

Format

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

`student`	a numeric vector
`teacher`	a numeric vector
`LCE`	a numeric vector
`PCE`	a numeric vector
`PVE`	a numeric vector
`VAE`	a numeric vector

See Also

`mlcfa.teacher`

Examples

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

`mlcfa.teacher` *XXM Dataset: mlcfa.teacher*

Description

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

Usage

```
data(mlcfa.teacher)
```

Format

A data frame with 163 observations on the following variable.

`teacher` a numeric vector

See Also

`mlcfa.student`

Examples

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

`pcwa.student` *XXM DATASET: 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

```
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

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

Format

A data frame with 93 observations on the following variable.

`teacher` a numeric vector

See Also

`pcwa.student`

Examples

```
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:

Branum-Martin, L. (in press) "Multilevel modeling: Practical examples to illustrate a special case of SEM" in Y. Petscher, C. Schatschneider, & D. L. Compton (Eds.) Applied Quantitative Analysis in the Social Sciences. New York: Routledge

This example illutsrates how a random-intercept model may be estiamted. 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()`.

```
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**

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

2. **teacher**

```
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.

```
 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 **team** 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**.

```
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**.

   ```
   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**.

   ```
   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:

```
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:

   `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`

`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

`supercoach` 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

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

Format

A data frame with 62 observations on the following variable.

`supercoach` a numeric vector

Examples

```
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

```
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) ...
```

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

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("te
   ```
2. **coach**
   ```
   threesixty <- xxmSubmodel(model = threesixty, level = "coach", parents = "supercoach", ys = c("coac
   ```
3. **supercoach**
   ```
   threesixty <- xxmSubmodel(model = threesixty, level = "supercoach", parents = , ys = , xs =, etas =c
   ```

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 variace 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_
   ```

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
   ```

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_pa
   ```

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, 
   ```

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,
   ```

4. **lamda**: 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_on_Super7** and **coach_on_Super8**). 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.

   ```
   threesixty <- xxmWithinMatrix(model = threesixty, level = "supercoach", type = "psi", pattern = ps3
   ```

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 indivators 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.

   `threesixty <- xxmBetweenMatrix(model = threesixty, parent = "supercoach", child = "teacher", type =`

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.

   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:

   `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`

`xxmBetweenMatrix` *Specifying Measurement, Structural and Covariance Relationships Across Levels*

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, `xxm` will generate informative labels for all parameters.
`name`	Name of the `xxmBetweenMatrix`. Name is optional. If no name is provided, `xxm` will generate an informative name.

Details

`xxmBetweenMatrix` 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 `xxmSubmodel` command when declaring the child level.

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

Value

`xxmBetweenMatrix` 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 `xxmBetweenMatrix` function is limited to directional type (i.e., type = "lambda"; type = "beta", type = "gamma"; type = "kappa").

See Also

`xxmModel`, `xxmSubmodel`, `xxmWithinMatrix`.

Examples

```
## Not run:
ly12_pat <- matrix(c(0,1,1,0,0,0,0,0,0,0,1,1),6,2)
ly12_val <- matrix(c(1.1,.9,.8,0,0,0,0,0,0,1,1.2,1.3),6,2)
mymodel <- xxmBetweenMatrix(model = mymodel, parent = "school", child = "student", type = "lambda", pattern = ly12_p

## End(Not run)
```

`xxmCI` *Estimate a Given xxm Model*

Description

`xxmCI` is used to estimate 95% Confidence Intervals for model parameters.

Usage

```
xxmCI(model)
```

Arguments

`model`	Name of the `xxmModel`.

Details

Once model parameters have been estimated by a call to `xxmRun`, profile CIs for the parameters may estimated by invoking `xxmCI`.

Value

As always, the function returns the model object.

Note

Model parameters must be estimated before calling `xxmCI`.

See Also

`xxmRun`

Examples

```
## Not run: xxmCI(model = mymodel)
```

`xxmGet` *Get information from an xxM model object.*

Description

Retreives useful information from an xxM model object.

Usage

```
xxmGet(model= , what=)
```

Arguments

`model`	Name of the `xxmModel`.
`what`	Current options are ("estimates", "fit").

Details

`xxmGet` is used to retreive useful information from a model object. Option what = "estimates" is used to retreive parameter estimates and CIs in a single table. Option what = "fit" is used to retreive fit information. Currently, only the ML deviance (-2LL) is returned.

Value

Returns appropriate requested value.

Note

`xxmGet` can only be invoked after model parameters have been estimated.

Examples

```
## Not run:
library(xxm)
est  <- xxmGet(model= xm, what="estimates")
deviance  <- xxmGet(model= xm, what="fit")

## End(Not run)
```

`xxmModel` *Create A New xxm Model*

Description

Creates 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 ued 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)
```

`xxmModelConstruction` `xxm` *Model: Parts and Whole*

Description

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 a 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`.

`xxmModelMatrices` *Model Matrices: What, Why and How*

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 specifing 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 refrerred 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 maningful 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 fist 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 `xxm` to "fix" the first element, and the value matrix tells `xxm` to fix the value to "1.0".

(3) Label matrix: A lavel matrix is used to provide informative labels to each parameter. There are no special rules for the choice of parameter labels and the matrix itslef is optional. If no label matrix is provided, `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 varible may be constructed as follows:

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 vale 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. `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)

Once the three component matrices are constructed the parameter matrix is created by invoking `xxmWithinMatrix` command:

myModel <- xxmWithinMatrix(model = myModel, level = "student", type = "lambda", pattern = lambda_pattern, value = lambda_value, label = lambda_lable, name = "StudentFactor-loading")

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:

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 vale 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. `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)

Once the three component matrices are constructed the parameter matrix is created by invoking `xxmBetweenMatrix` command:

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

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

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

`xxmSubmodel` *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 thant 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 inclde following columns: (1) ID column for the current level. Column name of the ID variable colmun must match the name of the curren 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` comand.

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

```
## 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)
```

`xxmSummary` *Retreives summary information from an xxM model object.*

Description

Retreives deviance and parameter estimates from an xxM model object.

Usage

```
xxmSummary(model= )
```

Arguments

`model`	Name of the `xxmModel`.

Details

`xxmSummary` is used to retreive useful information from a model object.

Value

Returns a list with two elements appropriate requested value.

`deviance`	Model deviance
`model`	Parameter estimates and 95% profile-likelihood confidence intervals

Note

`xxmSummary` can only be invoked after model parameters have been estimated.

Examples

```
## Not run:
library(xxm)
xm.summary  <- xxmSummary(model= xm)
deviance  <- xm.summary$deviance
estimates <-  xm.summary$estimates

## 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 refrerred 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, `xxm` will generate informative labels for all parameters.
`name`	Name of the `xxmWithinMatrix`. Name is optional. If no name is provided, `xxm` will generate an informative name.

Value

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

Note

1. Level must exists before the command can be invoked. This means that the level is declared in `xxmModel` and defined by invoking `xxmSubmodel`.
2. All levels must be defined using `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.

See Also

`xxmModel`, `xxmSubmodel`, `xxmBetweenMatrix`.

Examples

```
## Not run:
mymodel <- xxmWithinMatrix(model = mymodel, level = "student", type = "nu", pattern = nu1_pat, value = nu1_val, labe

## End(Not run)
```

Index