Package 'simsem'

October 1, 2011

Type Package

Title SIMulated Structural Equation Modeling data.

Version 0.0-1

Date 2011-09-22

Author@R c(person(``Sunthud'', ``Pornprasertmanit'', email = ``psunthud@ku.edu''))

Author Sunthud Pornprasertmanit (University of Kansas; psunthud@ku.edu)

Maintainer Sunthud Pornprasertmanit <psunthud@ku.edu>

Depends R(>= 2.12), methods, lavaan, MASS

Suggests OpenMx

Description This package will generate data for structural equation modeling framework. This package is tailored to use those simulated data for various purposes, such as model fit evaluation.

License GPL (>= 2)

LazyLoad yes

R topics documented:

`simsem-package` *SIMulated Structural Equation Modeling data.*

Description

This package will generate data for structural equation modeling framework. This package is tailored to use those simulated data for various purposes, such as model fit evaluation.

Details

Package:	simsem
Type:	Package
Version:	0.0.1
Depends:	R(>= 2.12), methods, lavaan, MASS
Suggests:	OpenMx
Date:	2011-09-22
License:	GPL (>= 2)
LazyLoad:	yes

Author(s)

Sunthud Pornprasertmanit (University of Kansas; psunthud@ku.edu)

Maintainer: Sunthud Pornprasertmanit (University of Kansas; psunthud@ku.edu)

`adjust.object` *Change an element in* `simMatrix`*,* `symMatrix`*, or* `simVector`*.*

Description

This function will adjust an element in `simMatrix`, `symMatrix`, or `simVector`. The specified element may be set to be free parameter with number or distribution object as starting values. The element can be fixed to be a value (such as 0).

Usage

```
adjust.object(target, simDist, position, constant.fixed)
```

Arguments

`target`	Target `simMatrix`, `symMatrix`, or `simVector` that you would like to adjusted.
`simDist`	The name of distribution object that you would like to specify (put as character with single or double quotation) or number that represents fixed values or starting values.
`position`	The position of element that you would like to adjust, such as `"c(1,2}"` is the row 1 and column 2 element of the specified matrix.
`constant.fixed`	This argument is used when the `simDist` item was specified as number. If true (as default), the number is treated as fixed parameters. If false, the number is treated as a starting value and free the parameter.

Value

Return the input `simMatrix`, `symMatrix`, or `simVector` with adjusted element.

Author(s)

Sunthud Pornprasertmanit (University of Kansas; psunthud@ku.edu)

See Also

`simMatrix` for random parameter matrix, `symMatrix` for symmetric random parameter matrix, and `simVector` for random parameter vector.

Examples

```
#loading <- matrix(0, 6, 2)
#loading[1:3, 1] <- NA
#loading[4:6, 2] <- NA
#LX <- matrix.object(loading, 0.7)
#summary(LX)
#run(LX)

#u34 <- runif.object(0.3, 0.4)
#LX <- adjust.object(LX, "u34", c(2, 1))
#summary(LX)
#run(LX)

#LX <- adjust.object(LX, 0, c(2,1))
#LX <- adjust.object(LX, 0.5, c(2,2), FALSE)
#summary(LX)
#run(LX)

#factor.mean <- rep(NA, 2)
#factor.mean.starting <- c(5, 2)
#AL <- vector.object(factor.mean, factor.mean.starting)
#run(AL)
#summary(AL)

#n01 <- rnorm.object(0, 1)
#AL <- adjust.object(AL, "n01", 2)
#run(AL)
#summary(AL)
```

---

combine.object *Combine two objects (Internal)*

Description

This function is used to combine two objects in the same or similar type together.

Usage

```
combine.object(object1, object2, ...)
```

Arguments

object1	The first object
object2	The second object
...	Additional options

Details

Candidate objects are vector, matrix, simMatrix, simVector, matrixSet, and misspecifiedSet

Value

The combined objects

Note

Internal Function

Author(s)

Sunthud Pornprasertmanit (University of Kansas; psunthud@ku.edu)

`constant.vector` *Create constant* `simVector` *(Internal)*

Description

Create a constant `simVector`

Usage

```
constant.vector(constant, ni)
```

Arguments

`constant`	Number that is used to be the constant
`ni`	Number of items

Value

Return constant `simVector`

Note

Internal function

Author(s)

Sunthud Pornprasertmanit (University of Kansas; psunthud@ku.edu)

Examples

```
#constant.vector(0, 4)
```

constrain.matrices *Impose equality constraint in an object (Internal)*

Description

Impose equality constraint in an object

Usage

```
constrain.matrices(object, simConstraint, ...)
```

Arguments

object	Desired object that would like to be constrained
simConstraint	simConstraint object specifying equality constraints
...	Other argumetns

Details

Candidate objects are class blankReducedMatrixSet. This class is used in freeParamSet, labelsSet, and reducedMatrixSet.

Value

The objects with equality constraints imposed

Note

Internal function

Author(s)

Sunthud Pornprasertmanit (University of Kansas; psunthud@ku.edu)

contain *Check whether an element is in a vector (Internal)*

Description

Check whether an element is in a vector

Usage

```
contain(element, Vector)
```

Arguments

element	Desired element that would like to be searched
Vector	Desired object matched with the element

Value

TRUE if the element is in the vector

Note

Internal Function

Author(s)

Sunthud Pornprasertmanit (University of Kansas; psunthud@ku.edu)

Examples

```
#contain(0, 1:3)
#contain(1, 1:3)
```

---

create.free.parameters

Create free parameters object from model specification

Description

Create free parameters object from model specification

Usage

```
create.free.parameters(object)
```

Arguments

`object`	`simMatrixSet` object

Value

`freeParamSet` object

Note

Internal Function

Author(s)

Sunthud Pornprasertmanit (University of Kansas; psunthud@ku.edu)

Examples

```
#loading <- matrix(0, 6, 2)
#loading[1:3, 1] <- NA
#loading[4:6, 2] <- NA
#loadingValues <- matrix(0, 6, 2)
#loadingValues[1:3, 1] <- 0.7
#loadingValues[4:6, 2] <- 0.7
#LX <- matrix.object(loading, loadingValues)

#latent.cor <- matrix(NA, 2, 2)
#diag(latent.cor) <- 1
#PH <- sym.matrix.object(latent.cor, 0.5)

#error.cor <- matrix(0, 6, 6)
#diag(error.cor) <- 1
#TD <- sym.matrix.object(error.cor)

#indicator.mean <- rep(NA, 6)
#MX <- vector.object(indicator.mean, 0)

#CFA.Model <- matrix.CFA.object(LX = LX, PH = PH, TD = TD, MX = MX)
#free <- create.free.parameters(CFA.Model)
```

---

create.implied.MACS

Create model implied Means and Covariance Matrix (MACS)

Description

Create model implied means and covariance matrix from a parameter set from any SEM model.

Usage

```
create.implied.MACS(object)
```

Arguments

object	`matrixSet` (both X and Y sides) or `reducedMatrixSet` (Y side only) that contains model parameters

Details

This function create model implied mean and covariance matrix by formulas.

Value

M	Model implied mean
CM	Model implied covariance matrix

Note

The equation is ... (TBA).

Author(s)

Sunthud Pornprasertmanit (University of Kansas; psunthud@ku.edu)

References

Ansari, A., Jedidi, K., & Jagpal, S. (2000). A hierarchical Bayesian methodology for treating heterogeneity in structural equation models. Marketing Science, 328-347.

Examples

```
#loading <- matrix(0, 6, 2)
#loading[1:3, 1] <- NA
#loading[4:6, 2] <- NA
#loadingValues <- matrix(0, 6, 2)
#loadingValues[1:3, 1] <- 0.7
#loadingValues[4:6, 2] <- 0.7
#LX <- matrix.object(loading, loadingValues)
#summary(LX)

#latent.cor <- matrix(NA, 2, 2)
#diag(latent.cor) <- 1
#PH <- sym.matrix.object(latent.cor, 0.5)

#error.cor <- matrix(0, 6, 6)
#diag(error.cor) <- 1
#TD <- sym.matrix.object(error.cor)

#CFA.Model <- matrix.CFA.object(LX = LX, PH = PH, TD = TD)
#CFA.Model.Param <- run(CFA.Model)
#create.implied.MACS(CFA.Model.Param)
```

---

divide.object *Make a division on each elements of the object (Internal)*

Description

Make a division on each elements of the object

Usage

```
divide.object(object, constant, ...)
```

Arguments

object	The desired object
constant	Divisor
...	Additional options

Details

Candidate objects are `vector`, `matrix`, and `matrixSet`

Value

The divided objects

Note

Internal Function

Author(s)

Sunthud Pornprasertmanit (University of Kansas; psunthud@ku.edu)

`find.OpenMx.values` *Rearrange starting values such that it is appropriate for OpenMx matrix specification (Internal)*

Description

Will combine both starting values and fixed value as the values command in OpenMx matrix

Usage

```
find.OpenMx.values(Parameters, Starting.Values)
```

Arguments

`Parameters`	Any objects that describe parameters.
`Starting.Values`	Any fixed values object that describe starting values

Value

Return object of constants that describe both starting values and fixed values.

Note

(Internal Function) Working for (`vector`, `vector`), (`matrix`, `matrix`), or (`freeParamSet`, `reducedMatrixSet`).

Author(s)

Sunthud Pornprasertmanit (University of Kansas; psunthud@ku.edu)

Examples

```
#parameter <- c(NA, NA, 0, 0)
#starting.values <- c(2, 5, 0, 0)
#find.OpenMx.Values(parameter, starting.values)
```

is.default *Check whether a specified* simVector *was a default* simVector *(Internal)*

Description

Check whether a specified simVector was a default simVector such that users did not specify anything. For example, check whether means of indicators are specified as 1.

Usage

```
is.default(object)
```

Arguments

object	checked simVector

Value

TRUE if the object is a default simVector. FALSE, otherwise.

Note

Internal Function

Author(s)

Sunthud Pornprasertmanit (University of Kansas; psunthud@ku.edu)

is.null.object *Check whether the object is NULL (Internal)*

Description

Check whether the object is NULL

Usage

```
is.null.object(target)
```

Arguments

target	Checked target

Value

TRUE if the object is null. FALSE otherwise.

Note

Internal Function

Author(s)

Sunthud Pornprasertmanit (University of Kansas; psunthud@ku.edu)

---

loading.from.alpha *Find standardized factor loading from coefficient alpha*

Description

Find standardized factor loading from coefficient alpha assuming that all items have equal loadings.

Usage

```
loading.from.alpha(alpha, ni)
```

Arguments

alpha	A desired coefficient alpha value.
ni	A desired number of items.

Value

result	The standardized factor loadings that make desired coefficient alpha with specified number of items.

Author(s)

Sunthud Pornprasertmanit (University of Kansas; psunthud@ku.edu)

Examples

```
loading.from.alpha(0.8, 4)
```

---

make.labels *Make parameter names for OpenMx (Internal)*

Description

Make parameter names for each element in a matrix or a vector for OpenMx syntax

Usage

```
make.labels(object, ...)
```

Arguments

object	the target objects
...	Name of the desired object, the analysis package, and the attribute told whether the object is symmetric

Value

Return the labels object

Note

Internal Function

Author(s)

Sunthud Pornprasertmanit (University of Kansas; psunthud@ku.edu)

`match.keyword` *Search for the keywords and check whether the specified text match one in the vector (Internal)*

Description

Search for the keywords and check whether the specified text match one in the vector

Usage

```
match.keyword(Names, keywords)
```

Arguments

`Names`	Name of the searching object
`keywords`	Name of the vector that would like to matched

Value

The position of keywords in the vector. 0 if the names does not match the specified vector.

Note

Internal Function

Author(s)

Sunthud Pornprasertmanit (University of Kansas; psunthud@ku.edu)

Examples

```
#match.keyword("LY", c("LY", "Ly", "ly", "LX", "Lx", "lx"))
```

`matrix.CFA.object` *Create a set of matrix that belongs to CFA model.*

Description

This function will create set of matrix that belongs to confirmatory factor analysis. The requirement is to specify factor loading matrix, factor correlation matrix, and error correlation matrix.

Usage

```
matrix.CFA.object(...)
```

Arguments

`...`	Each element of model specification, as described in `Details`

Details

NOTE: CFA object can be either specified in X or Y side. REQUIRED: `LX` or `LY` for factor loading matrix (need to be `simMatrix` object). REQUIRED: `TD` or `TE` for measurement error correlation matrix (need to be `symMatrix` object). REQUIRED: `PH` or `PH` for factor correlation matrix (need to be `symMatrix` object). `VTD` or `VTE` for measurement error variance (need to be `simVector` object). `VX` or `VY` for total indicator variance (need to be `simVector` object). NOTE: Either measurement error variance or indicator variance is specified. Both cannot be simultaneously specified. `VPH`, `VPS`, `VK`, or `VE` for factor total variance (need to be `simVector` object). NOTE: These four objects will have different meanings in `matrix.SEM.object` function. `TX` or `TY` for measurement intercepts. (need to be `simVector` object). `MX` or `MY` for overall indicator means. (need to be `simVector` object). NOTE: Either measurement intercept of indicator mean can be specified. Both cannot be specified simultaneously. `KA`, `AL`, `MK`, or `ME` for factor means (need to be `simVector` object).

DEFAULT: 1) All indicator variances are equal to 1. Measurement error variances are automatically implied from total indicator variances. 2) All measurement error variances are free parameters. 3) All indicator means are equal to 0. Indicator intercepts are automatically implied from indicator means. 4) All indicator intercepts are free parameters. 5) All factor variances are equal to 1. 6) All factor variances are fixed. 7) All factor means are equal to 0. 8) All factor means are fixed.

Value

`simMatrixSet` object that represents the CFA object. This will be used for specifying data or model objects later.

Author(s)

Sunthud Pornprasertmanit (University of Kansas; psunthud@ku.edu)

See Also

See class `simMatrixSet` for result object details. See `simMatrix`, `symMatrix`, or `simVector` for input details. Use `matrix.Path.object` to specify path analysis model and use `matrix.SEM.object` to specify full structural equation modeling.

Examples

```
loading <- matrix(0, 6, 2)
loading[1:3, 1] <- NA
loading[4:6, 2] <- NA
loadingValues <- matrix(0, 6, 2)
loadingValues[1:3, 1] <- 0.7
loadingValues[4:6, 2] <- 0.7
LX <- matrix.object(loading, loadingValues)
summary(LX)

latent.cor <- matrix(NA, 2, 2)
diag(latent.cor) <- 1
PH <- sym.matrix.object(latent.cor, 0.5)

# Error Correlation Object
error.cor <- matrix(0, 6, 6)
diag(error.cor) <- 1
TD <- sym.matrix.object(error.cor)

CFA.Model <- matrix.CFA.object(LX = LX, PH = PH, TD = TD)
```

`matrix.object` *Create matrix object that save free parameters and starting values, as well as fixed values*

Description

Create `simMatrix` object that save free parameters and starting values, as well as fixed values. This will be used for model specification later, such as for factor loading matrix or regression coefficient matrix.

Usage

```
matrix.object(Matrix, name.dist.object = NULL)
```

Arguments

`Matrix`	Matrix of free parameters. Use NA to specify free parameters. Use number as fixed value (including zero)
`name.dist.object`	Starting values. Can be either one element or matrix with the same dimension as free parameter matrix. Each element can be numbers (in either `as.numeric` or `as.character` format) or the name of distribution object `simDist`.

Value

`simMatrix` object that will be used for model specification later.

Author(s)

Sunthud Pornprasertmanit (University of Kansas; psunthud@ku.edu)

See Also

See simDist for the resulting object. See sym.matrix.object for creating symmetric matrix object and vector.object for vector object.

Examples

```
loading <- matrix(0, 6, 2)
loading[1:3, 1] <- NA
loading[4:6, 2] <- NA
loadingValues <- matrix(0, 6, 2)
loadingValues[1:3, 1] <- 0.7
loadingValues[4:6, 2] <- 0.7
LX <- matrix.object(loading, loadingValues)
summary(LX)
run(LX)

n65 <- rnorm.object(0.6, 0.05)
LY <- matrix.object(loading, "n65")
summary(LY)
run(LY)
```

---

matrix.Path.object *Create a set of matrix belongs to Path analysis model*

Description

This function will create set of matrix that belongs to path analysis model. The requirement is to specify indicator correlation and regression coefficient matrix.

Usage

```
matrix.Path.object(..., exo = FALSE)
```

Arguments

...	Each element of model specification, as described in Details
exo	specify TRUE if users wish to specify both exogenous and endogenous indicators.

Details

REQUIRED: BE for regression coefficient matrix (need to be simMatrix object). REQUIRED: PS for residual correlation matrix (need to be symMatrix object). VPS for residual indicator variance (need to be simVector object). VE for total indicator variance (need to be simVector object). NOTE: Either total indicator variance or residual indicator variance is specified. Both cannot be simultaneously specified. AL for indicator intercept (need to be simVector object). ME for indicator total mean (need to be simVector object). NOTE: Either indicator intercept or indicator total mean is specified. Both cannot be simultaneously specified.

VPS for residual indicator variance (need to be simVector object). VE for total indicator variance (need to be simVector object). NOTE: Either total indicator variance or residual indicator variance is specified. Both cannot be simultaneously specified. AL for indicator intercept (need to be

simVector object). ME for indicator total mean (need to be simVector object). NOTE: Either indicator intercept or indicator total mean is specified. Both cannot be simultaneously specified.

NOTE: If users need to specify exogenous variable too. REQUIRED for "exo=TRUE": GA for regression coefficient matrix from exogenous variable to endogenous variable (need to be simMatrix object). REQUIRED for "exo=TRUE": PH for exogenous factor correlation (need to be symMatrix object). VPH or VK for exogenous variable variance (need to be simVector object). KA or MK for exogenous variable mean (need to be simVector object).

DEFAULT: 1) All indicator variances are equal to 1. Residual variances are automatically implied from total indicator variances. 2) All residual variances are free parameters. 3) All indicator means are equal to 0. Intercepts are automatically implied from total indicator mean. 4) All indicator intercepts are free parameters.

Value

simMatrixSet object that represents the path analysis model object. This will be used for specifying data or model objects later.

Author(s)

Sunthud Pornprasertmanit (University of Kansas; psunthud@ku.edu)

See Also

See class simMatrixSet for result object details. See simMatrix, symMatrix, or simVector for input details. Use matrix.CFA.object to specify CFA model and use matrix.SEM.object to specify full structural equation modeling.

Examples

```
u35 <- runif.object(0.3, 0.5)
u57 <- runif.object(0.5, 0.7)
u1 <- runif.object(-0.1, 0.1)
n31 <- rnorm.object(0.3, 0.1)

path.BE <- matrix(0, 4, 4)
path.BE[3, 1:2] <- NA
path.BE[4, 3] <- NA
starting.BE <- matrix("", 4, 4)
starting.BE[3, 1:2] <- "u35"
starting.BE[4, 3] <- "u57"
BE <- matrix.object(path.BE, starting.BE)

residual.error <- diag(4)
residual.error[1,2] <- residual.error[2,1] <- NA
PS <- sym.matrix.object(residual.error, "n31")

Path.Model <- matrix.Path.object(PS = PS, BE = BE)

u35 <- runif.object(0.3, 0.5)
u57 <- runif.object(0.5, 0.7)
u1 <- runif.object(-0.1, 0.1)
n31 <- rnorm.object(0.3, 0.1)

path.GA <- matrix(0, 2, 2)
```

```
path.GA[1, 1:2] <- NA
GA <- matrix.object(path.GA, "u35")

path.BE <- matrix(0, 2, 2)
path.BE[2, 1] <- NA
BE <- matrix.object(path.BE, "u57")

exo.cor <- matrix(NA, 2, 2)
diag(exo.cor) <- 1
PH <- sym.matrix.object(exo.cor, "n31")

PS <- sym.matrix.object(diag(2))

Path.Exo.Model <- matrix.Path.object(PS = PS, BE = BE, PH = PH, GA = GA, exo=TRUE)
```

`matrix.SEM.object` *Create a set of matrix belongs to SEM model*

Description

This function will create set of matrix that belongs to full SEM model. The requirement is to specify factor residual correlation matrix, regression coefficient matrix, factor loading matrix, and measurement error correlation.

Usage

```
matrix.SEM.object(..., exo = FALSE)
```

Arguments

`...`	Each element of model specification, as described in `Details`
`exo`	specify TRUE if users wish to specify both exogenous and endogenous indicators.

Details

REQUIRED: `LY` for factor loading matrix from endogenous factors to Y indicators (need to be `simMatrix` object). REQUIRED: `TE` for measurement error correlation matrix among Y indicators (need to be `symMatrix` object). REQUIRED: `BE` for regression coefficient matrix among endogenous factors (need to be `simMatrix` object). REQUIRED: `PS` for residual correlation matrix among endogenous factors (need to be `symMatrix` object). `VTE` for measurement error variance of Y indicators (need to be `simVector` object). `VY` for total variance of Y indicators (need to be `simVector` object). NOTE: Either measurement error variance or indicator variance is specified. Both cannot be simultaneously specified. `TY` for measurement intercepts of Y indicators. (need to be `simVector` object). `MY` for overall Y indicator means. (need to be `simVector` object). NOTE: Either measurement intercept of indicator mean can be specified. Both cannot be specified simultaneously. `VPS` for residual variance of endogenous factors (need to be `simVector` object). `VE` for total endogenous factor variance (need to be `simVector` object). NOTE: Either total endogenous factor variance or residual endogenous factor variance is specified. Both cannot be simultaneously specified. `AL` for endogenous factor intercept (need to be `simVector` object). `ME` for total mean of endogenous factors (need to be `simVector` object). NOTE: Either endogenous

factor intercept or total mean of endogenous factor is specified. Both cannot be simultaneously specified.

NOTE: If users need to specify exogenous variable too. REQUIRED for "exo=TRUE": LX for factor loading matrix from exogenous factors to X indicators (need to be simMatrix object). REQUIRED for "exo=TRUE": TD for measurement error correlation matrix among X indicators (need to be symMatrix object). REQUIRED for "exo=TRUE": GA for regression coefficient matrix among exogenous factors (need to be simMatrix object). REQUIRED for "exo=TRUE": PH for residual correlation matrix among exogenous factors (need to be symMatrix object). VTD for measurement error variance of X indicators (need to be simVector object). VX for total variance of X indicators (need to be simVector object). NOTE: Either measurement error variance or indicator variance is specified. Both cannot be simultaneously specified. TX for measurement intercepts of Y indicators. (need to be simVector object). MX for overall Y indicator means. (need to be simVector object). NOTE: Either measurement intercept of indicator mean can be specified. Both cannot be specified simultaneously. VPH or VK for total exogenous factor variance (need to be simVector object). KA or MK for total mean of exogenous factors (need to be simVector object).

DEFAULT: 1) All indicator variances are equal to 1. Measurement error variances are automatically implied from total indicator variances. 2) All measurement error variances are free parameters. 3) All indicator means are equal to 0. Indicator intercepts are automatically implied from indicator means. 4) All indicator intercepts are free parameters. 5) All factor variances are equal to 1. 6) All factor variances are fixed. 7) All factor means are equal to 0. 8) All factor means are fixed.

Value

simMatrixSet object that represents the SEM object. This will be used for specifying data or model objects later.

Author(s)

Sunthud Pornprasertmanit (University of Kansas; psunthud@ku.edu)

See Also

See class simMatrixSet for result object details. See simMatrix, symMatrix, or simVector for input details. Use matrix.CFA.object to specify CFA model and use matrix.Path.object to specify path analysis model.

Examples

```
u68 <- runif.object(0.6, 0.8)
loading <- matrix(0, 8, 3)
loading[1:3, 1] <- NA
loading[4:6, 2] <- NA
loading[7:8, 3] <- NA
loading.start <- matrix("", 8, 3)
loading.start[1:3, 1] <- 0.7
loading.start[4:6, 2] <- 0.7
loading.start[7:8, 3] <- "u68"
LY <- matrix.object(loading, loading.start)

TE <- sym.matrix.object(diag(8))

factor.cor <- diag(3)
factor.cor[1, 2] <- factor.cor[2, 1] <- NA
```

```
PS <- sym.matrix.object(factor.cor, 0.5)

path <- matrix(0, 3, 3)
path[3, 1:2] <- NA
path.start <- matrix(0, 3, 3)
path.start[3, 1] <- "n65"
path.start[3, 2] <- "u35"
BE <- matrix.object(path, path.start)

SEM.model <- matrix.SEM.object(BE=BE, LY=LY, PS=PS, TE=TE)

loading.X <- matrix(0, 6, 2)
loading.X[1:3, 1] <- NA
loading.X[4:6, 2] <- NA
LX <- matrix.object(loading.X, 0.7)

loading.Y <- matrix(NA, 2, 1)
LY <- matrix.object(loading.Y, "u68")

TD <- sym.matrix.object(diag(6))

TE <- sym.matrix.object(diag(2))

factor.K.cor <- matrix(NA, 2, 2)
diag(factor.K.cor) <- 1
PH <- sym.matrix.object(factor.K.cor, 0.5)

PS <- sym.matrix.object(as.matrix(1))

path.GA <- matrix(NA, 1, 2)
path.GA.start <- matrix(c("n65", "u35"), ncol=2)
GA <- matrix.object(path.GA, path.GA.start)

BE <- matrix.object(as.matrix(0))

SEM.Exo.model <- matrix.SEM.object(GA=GA, BE=BE, LX=LX, LY=LY, PH=PH, PS=PS, TD=TD, TE=TE
```

`model.object` *Create model object from model specification*

Description

This function will take model specification from `simMatrixSet` that contains free parameters, starting values, and fixed values. It will transform the code to a specified SEM package and ready to analyze data.

Usage

```
model.object(object, ...)
```

Arguments

`object`	`simMatrixSet` or `freeParamSet` that provides model specification
`...`	Other values that will be explained specifically for each class

Value

`simModel` that will be used for data analysis

Author(s)

Sunthud Pornprasertmanit (University of Kansas; psunthud@ku.edu)

See Also

Each method link

Examples

```
#loading <- matrix(0, 6, 2)
#loading[1:3, 1] <- NA
#loading[4:6, 2] <- NA
#loadingValues <- matrix(0, 6, 2)
#loadingValues[1:3, 1] <- 0.7
#loadingValues[4:6, 2] <- 0.7
#LX <- matrix.object(loading, loadingValues)
#summary(LX)

#latent.cor <- matrix(NA, 2, 2)
#diag(latent.cor) <- 1
#PH <- sym.matrix.object(latent.cor, 0.5)

#error.cor <- matrix(0, 6, 6)
#diag(error.cor) <- 1
#TD <- sym.matrix.object(error.cor)

#CFA.Model <- matrix.CFA.object(LX = LX, PH = PH, TD = TD)

#SimModel <- model.object(CFA.Model)
```

`nullMatrix-class` *Class* `"nullMatrix"`

Description

Null Matrix (Internal)

Objects from the Class

Objects can be created by calls of the form `new("nullMatrix", ...)`.

Slots

`.Data`**:** No element in it

Methods

No methods defined with class "nullMatrix" in the signature.

Note

Internal Class

Author(s)

Sunthud Pornprasertmanit (University of Kansas; psunthud@ku.edu)

`nullSimMatrix-class`

Class Null object of `simMatrix`*,* `symMatrix`*, and* `simVector` *classes (Internal)*

Description

Represent null object of simMatrix class

Objects from the Class

Cannot create from user interface.

Slots

`Data`**:** Always NaN

`Labels`**:** Always NaN

Methods

run Return Null Matrix or Vector

Note

Internal Class

Author(s)

Sunthud Pornprasertmanit (University of Kansas; psunthud@ku.edu)

nullVector-class *Class* "nullVector"

Description

Null Vector (Internal)

Objects from the Class

Objects can be created by calls of the form `new("nullVector", ...)`.

Slots

`.Data`: No element in it

Extends

Class "vector", from data part.

Methods

No methods defined with class "nullVector" in the signature.

Note

Internal Class

Author(s)

Sunthud Pornprasertmanit (University of Kansas; psunthud@ku.edu)

print.if.not.null *Provide basic summary of each object if that object is not NULL (Internal)*

Description

Provide basic summary of each object if that object is not NULL (Internal)

Usage

```
print.if.not.null(object, name)
```

Arguments

`object`	Checked object whether it is NULL
`name`	Name of the object

Value

NONE. Will print on R screen only.

Note

Internal Function

Author(s)

Sunthud Pornprasertmanit (University of Kansas; psunthud@ku.edu)

Examples

```
#AL <- vector.object(rep(NA, 5), "0")
#print.if.not.null(AL, "Factor Mean")
```

Rnorm-class *Class "Rnorm"*

Description

Object that create a random number from normal distribution.

Objects from the Class

The object should be created by `rnorm.object` function. Objects can be created by calls of the form `new("Rnorm", ...)`.

Slots

`Mean`**:** Mean of the distribution

`SD`**:** Standard deviation of the distribution

Extends

Class "`simDist`", directly.

Methods

run `signature(object = "Rnorm")`: create a random number from the distribution

summary `signature(object = "Rnorm")`: summarize information in the object

Author(s)

Sunthud Pornprasertmanit (University of Kansas, psunthud@ku.edu)

Examples

```
showClass("Rnorm")
n2 <- rnorm.object(0, 0.2)
run(n2)
summary(n2)
```

rnorm.object *Create random normal distribution object*

Description

Create random normal distribution object. Random normal distribution object will save mean and standard deviation parameter. This will use in specifying parameters that distributed as normal distribution.

Usage

```
rnorm.object(Mean, SD)
```

Arguments

Mean	Desired population mean
SD	Desired population standard deviation

Value

Rnorm	Random Normal Distribution object that save the specified parameters

Author(s)

Sunthud Pornprasertmanit (University of Kansas; psunthud@ku.edu)

Examples

```
n02 <- rnorm.object(0, 0.2)
run(n02)
```

run *Run a particular object in simsem package.*

Description

Run a particular object such as running any distribution objects to create number.

Usage

```
run(object, ...)
```

Arguments

object	'simsem' object
...	any additional arguments

Value

object	depends on particular object

Author(s)

Sunthud Pornprasertmanit (University of Kansas; psunthud@ku.edu)

Examples

```
n02 <- rnorm.object(0, 0.2)
run(n02)
```

Runif-class *Class "Runif"*

Description

Object that create a random number from uniform distribution.

Objects from the Class

The object should be created by `runif.object` function. Objects can be created by calls of the form `new("Runif", ...)`.

Slots

`Lower`: Lower bound parameter

`Upper`: Upper bound parameter

Extends

Class "simDist", directly.

Methods

run `signature(object = "Runif")`: create a random number from the distribution

summary `signature(object = "Runif")`: summarize information in the object

Author(s)

Sunthud Pornprasertmanit (University of Kansas, psunthud@ku.edu)

Examples

```
showClass("Runif")
u1 <- runif.object(-0.1, 0.1)
run(u1)
summary(u1)
```

runif.object *Create random uniform distribution object*

Description

Create random uniform distribution object. Random uniform distribution object will save mean and standard deviation parameter. This will use in specifying parameters that distributed as normal distribution.

Usage

```
runif.object(Lower, Upper)
```

Arguments

Lower	Lower bound of the distribution
Upper	Upper bound of the distribution

Value

Runif	Random Uniform Distribution object that save the specified parameters

Author(s)

Sunthud Pornprasertmanit (University of Kansas; psunthud@ku.edu)

Examples

```
u1 <- runif.object(-0.1, 0.1)
run(u1)
```

simDist-class *Class "simDist"*

Description

All distribution objects

Objects from the Class

A virtual Class: No objects may be created from it.

Methods

No methods defined with class "simDist" in the signature.

Author(s)

Sunthud Pornprasertmanit (University of Kansas, psunthud@ku.edu)

Examples

```
showClass("simDist")
```

simMatrix-class *Class* "simMatrix" *(Random parameters matrix)*

Description

This object can be used to represent a matrix in SEM model. It contains free parameters, fixed values, and starting values. This object can be represented factor loading matrix or regreesion coefficient matrix.

Objects from the Class

This object

Slots

`Data`**:** indicates which elements of the matrix are free or fixed. "NA" means the element is freely estimated. Numbers (including 0) means the element is fixed to be the indicated number.

`Labels`**:** indicates the starting values of each element in the matrix. The starting values could be numbers or the name of

Methods

adjust.object `signature(target = "simMatrix")`: adjust an element in the "simMatrix" object

run `signature(object = "simMatrix")`: draws starting values from the "labels" slot and show as a matrix sample.

summary.short `signature(object = "simMatrix")`: provides a short summary of all information in the object

summary `signature(object = "simMatrix")`: provides a thorough description of all information in the object

Author(s)

Sunthud Pornprasertmanit (University of Kansas; psunthud@ku.edu)

See Also

later

Examples

```
showClass("simMatrix")

#loading <- matrix(0, 6, 2)
#loading[1:3, 1] <- NA
#loading[4:6, 2] <- NA
#loadingValues <- matrix(0, 6, 2)
#loadingValues[1:3, 1] <- 0.7
#loadingValues[4:6, 2] <- 0.7
#LX <- matrix.object(loading, loadingValues)
#summary(LX)
```

```
#run(LX)

#n65 <- rnorm.object(0.6, 0.05)
#LY <- matrix.object(loading, "n65")
#summary(LY)
#run(LY)

#u34 <- runif.object(0.3, 0.4)
#LY <- adjust.object(LY, "u34", c(2, 1))
#summary(LY)
#run(LY)
#summary.short(LY)
```

---

simMatrixSet-class *Class* "simMatrixSet"

Description

Set of vectors and matrices that saves model specification (CFA, Path analysis, or SEM)

Objects from the Class

Object can be created by `matrix.Path.object`, `matrix.Path.object`, or `matrix.Path.object`, for CFA, Path analysis, or SEM model, respectively. Objects can be also created by calls of the form `new("simMatrixSet", ...)`.

Slots

`Tag`: Model type (CFA, Path, or SEM)

`LY`: Factor loading matrix between endogenous factors and Y indicators

`TE`: Correlation matrix between Y measurement error

`VTE`: Variance of Y measurement error

`PS`: Residual correlation of endogenous factors

`VPS`: Residual variances of endogenous factors

`BE`: Regression effect among endogenous factors

`TY`: Measurement intercepts of Y indicators

`AL`: Factor intercepts of endogenous factors

`ME`: Factor means of endogenous factors

`MY`: Total Mean of Y indicators

`VE`: Total variance of endogenous factors

`VY`: Total variance of Y indicators

`LX`: Factor loading matrix between exogenous factors and X indicators

`TD`: Correlation matrix between X measurement error

`VTD`: Variance of X measurement error

`PH`: Correlation among exogenous factors

`GA`: Regreeion effect from exogenous factors to endogenous factors

`TX`: Measurement intercepts of X indicators

KA: Factor Mean of exogenous factors

MX: Total Mean of X indicators

VPH: Variance of exogenous factors

VX: Total variance of X indicators

TH: Measurement error correlation between X indicators and Y indicators

Methods

summary Get the summary of model specification

Author(s)

Sunthud Pornprasertmanit (University of Kansas; psunthud@ku.edu)

See Also

Create this class by CFA, Path Analysis, or SEM model by `matrix.Path.object`, `matrix.Path.object`, or `matrix.Path.object`, respectively.

Examples

```
showClass("simMatrixSet")

loading <- matrix(0, 6, 2)
loading[1:3, 1] <- NA
loading[4:6, 2] <- NA
loadingValues <- matrix(0, 6, 2)
loadingValues[1:3, 1] <- 0.7
loadingValues[4:6, 2] <- 0.7
LX <- matrix.object(loading, loadingValues)
summary(LX)

latent.cor <- matrix(NA, 2, 2)
diag(latent.cor) <- 1
PH <- sym.matrix.object(latent.cor, 0.5)

# Error Correlation Object
error.cor <- matrix(0, 6, 6)
diag(error.cor) <- 1
TD <- sym.matrix.object(error.cor)

CFA.Model <- matrix.CFA.object(LX = LX, PH = PH, TD = TD)
summary(CFA.Model)
#run(CFA.Model)
```

---

simVector-class *Class* "simVector" *(Random parameters vector)*

Description

This object can be used to represent a vector in SEM model. It contains free parameters, fixed values, and starting values. This object can be represented mean, intercept, or variance vectors.

Objects from the Class

This object is created by `vector.object` function. Objects can be created by calls of the form `new("simVector", ...)`.

Slots

`Data`: Object of class `"vector"` draws starting values from the `"labels"` slot and show as a vector sample.

`Labels`: Object of class `"vector"` provides a thorough description of all information in the object

Methods

adjust.object `signature(target = "simVector")`: adjust an element in the `"simVector"` object

run `signature(object = "simVector")`: draws starting values from the `"labels"` slot and show as a vector sample.

summary.short `signature(object = "simVector")`: provides a short summary of all information in the object

summary `signature(object = "simVector")`: provides a thorough description of all information in the object

Author(s)

Sunthud Pornprasertmanit (University of Kansas; psunthud@ku.edu)

See Also

`simMatrix` for random parameter matrix and `symMatrix` for random parameter symmetric matrix.

Examples

```
showClass("simVector")

factor.mean <- rep(NA, 2)
factor.mean.starting <- c(5, 2)
AL <- vector.object(factor.mean, factor.mean.starting)
run(AL)
summary(AL)
summary.short(AL)

n01 <- rnorm.object(0, 1)
AL <- adjust.object(AL, "n01", 2)
run(AL)
summary(AL)
```

`starting.values` *Find starting values of free parameters (Internal)*

Description

Find starting values of free parameters based on pre-specified starting values. If the pre-specified starting values are numbers, the function will use that values. If they are distribution object, this function will randomly draw from the distribution 10 times and take the average of those values.

Usage

```
starting.values(object, trial, ...)
```

Arguments

`object`	A specified `simMatrix`, `simVector`, or `simMatrixSet` that wish to find starting values
`trial`	Number of random drawn to find starting values of distribution objects
`...`	Other arguments

Details

This function can be used for `simMatrix`, `simVector`, and `simMatrixSet`.

Value

`matrix`, `vector`, or `matrixSet` of starting values

Note

Internal Function

Author(s)

Sunthud Pornprasertmanit (University of Kansas; psunthud@ku.edu)

Examples

```
#u89 <- runif.object(0.8, 0.9)
#loading <- matrix(0, 6, 2)
#loading[1:3, 1] <- NA
#loading[4:6, 2] <- NA
#loadingValues <- matrix(0, 6, 2)
#LX <- matrix.object(loading, "u89")

#latent.cor <- matrix(NA, 2, 2)
#diag(latent.cor) <- 1
#PH <- sym.matrix.object(latent.cor, 0.5)

#error.cor <- matrix(0, 6, 6)
#diag(error.cor) <- 1
#TD <- sym.matrix.object(error.cor)
```

```
#CFA.Model <- matrix.CFA.object(LX = LX, PH = PH, TD = TD)
#starting.values(LX, 10)
#result <- starting.values(CFA.Model, 10)
#summary(result)
```

`summary.short` *Provide short summary of an object.*

Description

Provide short summary if it is available. Otherwise, it is an alias for `summary`.

Usage

```
summary.short(object, ...)
```

Arguments

`object`	Desired object being described
`...`	any additional arguments

Value

NONE. This function will print on screen only.

Author(s)

Sunthud Pornprasertmanit (University of Kansas; psunthud@ku.edu)

See Also

See help file on each class for details of `summary.short` function in each class

Examples

```
#u89 <- runif.object(0.8, 0.9)
#loading <- matrix(0, 6, 2)
#loading[1:3, 1] <- NA
#loading[4:6, 2] <- NA
#loadingValues <- matrix(0, 6, 2)
#LX <- matrix.object(loading, "u89")
#summary.short(LX)
```

`sym.matrix.object` *Create symmetric matrix object that save free parameters and starting values, as well as fixed values*

Description

Create `symMatrix` object that save free parameters and starting values, as well as fixed values. This will be used for model specification later, such as for factor residual correlation matrix or measurement error correlation matrix.

Usage

```
sym.matrix.object(Matrix, name.dist.object = NULL)
```

Arguments

`Matrix`	Symmetric matrix of free parameters. Use NA to specify free parameters. Use number as fixed value (including zero). The input matrix need to be symmetric matrix.
`name.dist.object`	Starting values. Can be either one element or matrix with the same dimension as free parameter matrix. Each element can be numbers (in either `as.numeric` or `as.character` format) or the name of distribution object `simDist`.

Value

`symMatrix` object that will be used for model specification later.

Author(s)

Sunthud Pornprasertmanit (University of Kansas; psunthud@ku.edu)

See Also

See `simDist` for the resulting object. See `matrix.object` for creating matrix object and `vector.object` for vector object.

Examples

```
latent.cor <- matrix(NA, 3, 3)
diag(latent.cor) <- 1
PH <- sym.matrix.object(latent.cor, 0.5)

u46 <- runif.object(0.4, 0.6)
factor.cor <- matrix(NA, 4, 4)
diag(factor.cor) <- 1
factor.cor.start <- matrix("u46", 4, 4)
factor.cor.start[1, 2] <- factor.cor.start[2, 1] <- "0.5"
PS <- sym.matrix.object(factor.cor, factor.cor.start)
```

symMatrix-class *Class* "symMatrix" *(Random parameters symmetric matrix)*

Description

This object can be used to represent a symmetric matrix in SEM model. It contains free parameters, fixed values, and starting values. This object can be represented factor correlation or error correlation matrix.

Objects from the Class

This object is created by "sym.matrix.object" function. Objects can be also created by calls of the form new("symMatrix", ...).

Slots

Data: indicates which elements of the matrix are free or fixed. "NA" means the element is freely estimated. Numbers (including 0) means the element is fixed to be the indicated number.

Labels: indicates the starting values of each element in the matrix. The starting values could be numbers or the name of "distribution objects"

Extends

Class "simMatrix", directly.

Methods

run signature(object = "symMatrix"): draws starting values from the "labels" slot and show as a symmetric matrix sample.

summary signature(object = "symMatrix"): provides a thorough description of all information in the object

Author(s)

Sunthud Pornprasertmanit (University of Kansas; psunthud@ku.edu)

See Also

simMatrix for random parameter matrix and simVector for random parameter vector.

Examples

```
showClass("symMatrix")

latent.cor <- matrix(NA, 3, 3)
diag(latent.cor) <- 1
PH <- sym.matrix.object(latent.cor, 0.5)

u46 <- runif.object(0.4, 0.6)
PH <- adjust.object(PH, "u46", c(3,2))
summary(PH)
summary.short(PH)
run(PH)
```

tag.headers *Name each element of specified matrices or vectors (Internal)*

Description

This element will add names in each element of a vector or will add row and columns names of a matrix with variable or factor names

Usage

```
tag.headers(object, ...)
```

Arguments

object	blankReducedMatrixSet which will be inherited by freeParamSet, labelsSet, and reducedMatrixSet
...	Other arguments

Details

Y means indicators on Y-side. X means indicators on X-side. E means endogenous factors. K means exogenous factors.

Value

Object with tags on it.

Note

Internal Function

Author(s)

Sunthud Pornprasertmanit (University of Kansas; psunthud@ku.edu)

vector.object *Create vector object that save free parameters and starting values, as well as fixed values*

Description

Create simVector object that save free parameters and starting values, as well as fixed values. This will be used for model specification later, such as for factor mean vector or measurement error variance vector.

Usage

```
vector.object(Vector, name.dist.object = NULL)
```

Arguments

`Vector`	Vector of free parameters. Use NA to specify free parameters. Use number as fixed value (including zero).
`name.dist.object`	Starting values. Can be either one element or vector with the same length as free parameter vector. Each element can be numbers (in either `as.numeric` or `as.character` format) or the name of distribution object `simDist`.

Value

`simVector` object that will be used for model specification later.

Author(s)

Sunthud Pornprasertmanit (University of Kansas; psunthud@ku.edu)

See Also

See `simDist` for the resulting object. See `matrix.object` for creating matrix object and `sym.matrix.object` for symmetric matrix object.

Examples

```
factor.mean <- rep(NA, 4)
AL <- vector.object(factor.mean, 0)

n02 <- rnorm.object(0, 0.2)
factor.start <- rep("n02", 4)
KA <- vector.object(factor.mean, factor.start)
```

Index