Package ‘simsem’

September 30, 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

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 NA

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
Date:	2011-09-22
License:	NA
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)

`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)

`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)
```

`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.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)

`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)
```

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

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"

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

symMatrix for symmetric random parameter matrix and simVector for random parameter vector.

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)
```

`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

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)
```

`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)

Index