Package 'simsem'

October 10, 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. Alternatively, 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 adjust.
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)" for the element in Row 1 and Column 2 in the specified matrix.
constant.fixed	This argument is used when the simDist argument 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 the element is set to be free parameter.

Value

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

Note

For symMatrix class, above- and below-diagonal elements will be adjusted simultaneously. Either above- or below-diagonal element is specified in the position argument.

Author(s)

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

See Also

1. simMatrix for random parameter matrix
2. symMatrix for symmetric random parameter matrix
3. 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)
```

---

constraint.object *Equality Constraint Object*

Description

This function will be used to specify equality constraints.

Usage

```
constraint.object(..., Tag)
```

Arguments

`...`	Each equality constraint in the model will be specified as a matrix. Rows represent elements that users wish to constrain. For single-group analysis, two columns are needed in the matrix. The first column indicates row of elements and second columns indicates columns of elements. Rownames will represent the matrix of elements that they are in. The detail section will discuss about how to specify row names. The first example shown below will show how to specify equality constraints for `LY(1,1)`, `LY(2,1)`, and `LY(3,1)`. For multiple groups, the columns will be three instead. The first column represent groups. The second and third columns represent row and column, respectively. The second example shown below will show how to specify equality constraints for `BE(2, 1)` of two groups. If you have multiple equality constriants, you can

make multiple matrices to represent them and add in the function. See the third example for multiple constraints.

`Tag` Type of analysis: `CFA`, `Path`, `Path.exo`, `SEM`, or `SEM.exo`.

Details

Row names specification depends on type of model. If users specify CFA model, the specification in shown in `matrix.CFA.object` function. If users specify Path analysis with or without exogenous variables, the specification is shown in `matrix.Path.object` function. If users specify SEM model with or without exogenous variables, the specification is shown in `matrix.SEM.object` function. However, basically, the names of matrices you put in these function are also eligible for this function as well.

Value

Object in `simConstraint` that save those equality constraints.

Note

The available constraints now are equality constraints. We expect to create nonlinear constraints soon.

Author(s)

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

See Also

1. `matrix.CFA.object` to see model specification in CFA model
2. `matrix.Path.object` to see model specification in Path analysis model
3. `matrix.SEM.object` to see model specification in SEM model
4. `simConstraint` for the result object

Examples

```
# Example 1: Single-group, one constraint
constraint <- matrix(0, 3, 2)
constraint[1,] <- c(1, 1)
constraint[2,] <- c(2, 1)
constraint[3,] <- c(3, 1)
rownames(constraint) <- rep("LY", 3)
equal.loading <- constraint.object(constraint, Tag="SEM.exo")

# Example 2: Multiple-group, one constraint
group.con <- matrix(0, 2, 3)
group.con[1,] <- c(1, 2, 1)
group.con[2,] <- c(2, 2, 1)
rownames(group.con) <- rep("BE", 2)
equal.path <- constraint.object(group.con, Tag="Path")

# Example 3: Single-group, multiple constraints
constraint1 <- matrix(1, 3, 2)
constraint1[,1] <- 1:3
rownames(constraint1) <- rep("LY", 3)
```

```
constraint2 <- matrix(2, 3, 2)
constraint2[,1] <- 4:6
rownames(constraint2) <- rep("LY", 3)
constraint3 <- matrix(3, 2, 2)
constraint3[,1] <- 7:8
rownames(constraint3) <- rep("LY", 2)
equal.loading2 <- constraint.object(constraint1, constraint2, constraint3, Tag="SEM")
summary(equal.loading2)
```

`data.object` *Data object*

Description

This function will be used to create data specification and ready for data simulation.

Usage

```
data.object(N, simMatrixSet, simMisspecifiedSet = new("nullSimMisspecifiedSet"),
```

Arguments

`N`	Desired sample size
`simMatrixSet`	Model specification matrices that are created by `matrix.CFA.object`, `matrix.Path.object` or `matrix.SEM.object`.
`simMisspecifiedSet`	Model *misspecification* matrices that are created by `misspecified.CFA.object`, `misspecified.Path.object`, or `misspecified.SEM.object`.
`simConstraint`	Equality constraints that are created by `constraint.object`. This will specify equality econstraints of parameters in data generation process.
`Constrain.Parameters.Only`	`TRUE` if users wish to constrain parameters before adding misspecification. `FALSE` if users wish to constrain parameters after adding misspecification.
`Misfit.bound`	Upper bound of population root mean squared error of approximation (RMSEA; Browne & Cudeck, 1992) that users wish their model misspecification to be
`Maximum.random`	The maximum number of random drawn parameters and misspecification model until all parameters in the model are eligible (no negative error variance, standardized coefficients over 1).

Details

This function will use `mvrnorm` function in `MASS` package to create data from model implied covariance matrix.

Value

simData object that save data model specification.

Author(s)

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

References

- Stieger, J. H. & Lind, J. C. (1980). *Statistically based tests for the number of factors*. Paper presented at the annual spring meeting of the Psychometric Society, Iowa City, IA.
- Browne, M. W., & Cudeck, R. (1992). Alternative ways of assessing model fit. *Sociological Methods & Research, 21*, 230-258.

See Also

1. matrix.CFA.object to see CFA model specification
2. matrix.Path.object to see Path analysis model specification
3. matrix.SEM.object to see SEM model specification
4. misspecified.CFA.object for specifying misspecification in CFA model
5. misspecified.Path.object for specifying misspecification in Path analysis model
6. misspecified.SEM.object for specifying misspecification in SEM model
7. constraint.object for setting equality constraints.

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)
CFA.Model <- matrix.CFA.object(LY = LX, PS = PH, TE = TD)
SimData <- data.object(200, CFA.Model)
summary(SimData)
run(SimData)

# With Misspecification Model
n01 <- rnorm.object(0, 0.1)
error.cor.Mis <- matrix(NA, 6, 6)
diag(error.cor.Mis) <- 1
TD.Mis <- sym.matrix.object(error.cor.Mis, "n01")
CFA.Model.Mis <- misspecified.CFA.object(TD=TD.Mis)
SimData <- data.object(200, CFA.Model, simMisspecifiedSet=CFA.Model.Mis)
summary(SimData)
run(SimData)
```

`find.cutoff` *Find cutoff given a priori alpha level*

Description

Extract fit indices information from the `simResult` and find cutoff of fit indices given a priori alpha level

Usage

```
find.cutoff(object, alpha, reverse=FALSE, used.fit=NULL)
```

Arguments

`object`	`simResult` that saves the analysis results from multiple replications
`alpha`	A priori alpha level
`reverse`	The default is to find criticl point on the side that indicates worse fit (the right side of RMSEA or the left side of CFI). If specifying as `TRUE`, the directions are reversed.
`used.fit`	Vector of names of fit indices that researchers wish to find cutoffs from. The default is to find cutoffs of all fit indices.

Value

One-tailed cutoffs of several fit indices with a priori alpha level

Author(s)

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

See Also

`simResult` for a detail of result 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)
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(LY = LX, PS = PH, TE = TD)
SimData <- data.object(200, CFA.Model)
SimModel <- model.object(CFA.Model)
```

```
# We make the examples running only 50 replications to save time.
# In reality, more replications are needed.
Output <- result.object(SimData, SimModel, 50)
find.cutoff(Output, 0.05)
```

`find.power` *Find power in rejecting alternative models based on fit indices criteria*

Description

Find the proportion of fit indices that indicate worse fit than a specified cutoffs. The cutoffs may be calculated from `find.cutoff` of the null model.

Usage

```
find.power(object.alt, cutoff, reverse = FALSE, used.fit=NULL)
```

Arguments

`object.alt`	`simResult` that indicates alternative model that users wish to reject
`cutoff`	Fit indices cutoffs from null model or users. This should be a vector with a specified fit indices names as the name of vector elements. The best way to specify cutoff is to calculate from `find.cutoff` function.
`reverse`	The default is to count the proportion of fit indices that indicates lower fit to the model, such as how many RMSEA in the alternative model that is worse than cutoffs. The direction can be reversed by setting as `TRUE`.
`used.fit`	Vector of names of fit indices that researchers wish to find cutoffs from. The default is to find cutoffs of all fit indices.

Value

List of power given different fit indices.

Author(s)

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

See Also

1. `find.cutoff` to find the cutoffs from null model.
2. `simResult` to see how to create result object

Examples

```
loading.null <- matrix(0, 6, 1)
loading.null[1:6, 1] <- NA
LX.NULL <- matrix.object(loading.null, 0.7)
PH.NULL <- sym.matrix.object(diag(1))
TD <- sym.matrix.object(diag(6))
CFA.Model.NULL <- matrix.CFA.object(LY = LX.NULL, PS = PH.NULL, TE = TD)
SimData.NULL <- data.object(500, CFA.Model.NULL)
SimModel <- model.object(CFA.Model.NULL)
```

```
# We make the examples running only 50 replications to save time.
# In reality, more replications are needed.
Output.NULL <- result.object(SimData.NULL, SimModel, 50)
Cut.NULL <- find.cutoff(Output.NULL, 0.95)

u79 <- runif.object(0.7, 0.9)
loading.alt <- matrix(0, 6, 2)
loading.alt[1:3, 1] <- NA
loading.alt[4:6, 2] <- NA
LX.ALT <- matrix.object(loading.alt, 0.7)
latent.cor.alt <- matrix(NA, 2, 2)
diag(latent.cor.alt) <- 1
PH.ALT <- sym.matrix.object(latent.cor.alt, "u79")
CFA.Model.ALT <- matrix.CFA.object(LY = LX.ALT, PS = PH.ALT, TE = TD)
SimData.ALT <- data.object(500, CFA.Model.ALT)
Output.ALT <- result.object(SimData.ALT, SimModel, 50)
find.power(Output.ALT, Cut.NULL)
Rule.of.thumb <- c(RMSEA=0.05, CFI=0.95, TLI=0.95, SRMR=0.06)
find.power(Output.ALT, Rule.of.thumb, used.fit=c("RMSEA", "CFI", "TLI", "SRMR"))
```

`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)
```

`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.

1. REQUIRED: `LX` or `LY` for factor loading matrix (need to be `simMatrix` object).
2. REQUIRED: `TD` or `TE` for measurement error correlation matrix (need to be `symMatrix` object).
3. REQUIRED: `PH` or `PH` for factor correlation matrix (need to be `symMatrix` object).
4. `VTD` or `VTE` for measurement error variance (need to be `simVector` object).
5. `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.
6. `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.
7. `TX` or `TY` for measurement intercepts. (need to be `simVector` object).
8. `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.
9. `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

1. See class simMatrixSet for result object details.
2. See simMatrix, symMatrix, or simVector for input details.
3. 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.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

1. See simDist for the resulting object.
2. See sym.matrix.object for creating symmetric matrix object.
3. See 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

The matrices and vectors in the endogenous side are

1. REQUIRED: BE for regression coefficient matrix (need to be simMatrix object).
2. REQUIRED: PS for residual correlation matrix (need to be symMatrix object).
3. VPS for residual indicator variance (need to be simVector object).
4. 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.
5. AL for indicator intercept (need to be simVector object).
6. 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.
7. VPS for residual indicator variance (need to be simVector object).
8. 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.
9. AL for indicator intercept (need to be simVector object).
10. 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.

If users wish to include the exogenous side in their models, these options are available,

1. REQUIRED for "exo=TRUE": GA for regression coefficient matrix from exogenous variable to endogenous variable (need to be simMatrix object).
2. REQUIRED for "exo=TRUE": PH for exogenous factor correlation (need to be symMatrix object).
3. VPH or VK for exogenous variable variance (need to be simVector object).
4. KA or MK for exogenous variable mean (need to be simVector object). NOTE: Either total indicator variance or residual indicator variance is specified. Both cannot be simultaneously specified.

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

1. See class simMatrixSet for result object details.
2. See simMatrix, symMatrix, or simVector for input details.
3. 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

The matrices and vectors in the endogenous side are

1. REQUIRED: `LY` for factor loading matrix from endogenous factors to Y indicators (need to be `simMatrix` object).
2. REQUIRED: `TE` for measurement error correlation matrix among Y indicators (need to be `symMatrix` object).
3. REQUIRED: `BE` for regression coefficient matrix among endogenous factors (need to be `simMatrix` object).
4. REQUIRED: `PS` for residual correlation matrix among endogenous factors (need to be `symMatrix` object).
5. `VTE` for measurement error variance of Y indicators (need to be `simVector` object).
6. `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.
7. `TY` for measurement intercepts of Y indicators. (need to be `simVector` object).
8. `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.
9. `VPS` for residual variance of endogenous factors (need to be `simVector` object).
10. `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.
11. `AL` for endogenous factor intercept (need to be `simVector` object).
12. `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.

If users need to specify exogenous variable too, these matrices and vectors are available,

1. REQUIRED for "exo=TRUE": LX for factor loading matrix from exogenous factors to X indicators (need to be simMatrix object).
2. REQUIRED for "exo=TRUE": TD for measurement error correlation matrix among X indicators (need to be symMatrix object).
3. REQUIRED for "exo=TRUE": GA for regression coefficient matrix among exogenous factors (need to be simMatrix object).
4. REQUIRED for "exo=TRUE": PH for residual correlation matrix among exogenous factors (need to be symMatrix object).
5. VTD for measurement error variance of X indicators (need to be simVector object).
6. 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.
7. TX for measurement intercepts of Y indicators. (need to be simVector object).
8. 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.
9. VPH or VK for total exogenous factor variance (need to be simVector object).
10. 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

1. See class simMatrixSet for result object details.
2. See simMatrix, symMatrix, or simVector for input details.
3. 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
```

misspecified.CFA.object
Set of model misspecification for CFA model.

Description

This function will define model misspecifiction from a defined model. This function is similar to matrix.CFA.object such that the matrices that indicates misspecification will be added as arguments in the function. However, users do not have to add all matrices and vectors in the function. Only element indicating misspecification is added.

Usage

```
misspecified.CFA.object(...)
```

Arguments

...	Arguments definition is listed in the Details section of matrix.CFA.object. Again, this function does not require to list all required matrices or vectors like the matrix.CFA.object function. Only misspecification is added.

Value

object in simMisspecifiedSet that saves model misspecification.

Author(s)

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

See Also

1. matrix.CFA.object for matrix definition and how to specify CFA model
2. simMisspecifiedSet for the result object
3. misspecified.Path.object for misspecification model in Path analysis and misspecified.SEM.obje for misspecification model in SEM.

Examples

```
n01 <- rnorm.object(0, 0.1)
error.cor.Mis <- matrix(NA, 6, 6)
diag(error.cor.Mis) <- 1
TD.Mis <- sym.matrix.object(error.cor.Mis, "n01")
CFA.Model.Mis <- misspecified.CFA.object(TD=TD.Mis)
```

misspecified.Path.object *Set of model misspecification for Path analysis model.*

Description

This function will define model misspecifiction from a defined model. This function is similar to matrix.Path.object such that the matrices that indicates misspecification will be added as arguments in the function. However, users do not have to add all matrices and vectors in the function. Only element indicating misspecification is added.

Usage

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

Arguments

...	Arguments definition is listed in the Details section of matrix.Path.object. Again, this function does not require to list all required matrices or vectors like the matrix.Path.object function. Only misspecification is added.
exo	specify TRUE if users wish to specify both exogenous and endogenous indicators.

Value

object in simMisspecifiedSet that saves model misspecification.

Author(s)

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

See Also

1. matrix.Path.object for matrix definition and how to specify Path analysis model
2. simMisspecifiedSet for the result object
3. misspecified.CFA.object for misspecification model in CFA and misspecified.SEM.object for misspecification model in SEM.

Examples

```
u1 <- runif.object(-0.1, 0.1)
mis.path.GA <- matrix(0, 2, 2)
mis.path.GA[2, 1:2] <- NA
mis.GA <- matrix.object(mis.path.GA, "u1")
Path.Mis.Model <- misspecified.Path.object(GA = mis.GA, exo=TRUE)
```

misspecified.SEM.object
Set of model misspecification for SEM model.

Description

This function will define model misspecifiction from a defined model. This function is similar to matrix.SEM.object such that the matrices that indicates misspecification will be added as arguments in the function. However, users do not have to add all matrices and vectors in the function. Only element indicating misspecification is added.

Usage

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

Arguments

...	Arguments definition is listed in the Details section of matrix.SEM.object. Again, this function does not require to list all required matrices or vectors like the matrix.SEM.object function. Only misspecification is added.
exo	specify TRUE if users wish to specify both exogenous and endogenous indicators.

Value

object in simMisspecifiedSet that saves model misspecification.

Author(s)

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

See Also

1. matrix.SEM.object for matrix definition and how to specify SEM model
2. simMisspecifiedSet for the result object
3. misspecified.CFA.object for misspecification model in CFA and misspecified.Path.object for misspecification model in Path analysis.

Examples

```
u2 <- runif.object(-0.2, 0.2)
n1 <- rnorm.object(0, 0.1)
loading.X.trivial <- matrix(NA, 6, 2)
loading.X.trivial[is.na(loading.X.trivial)] <- 0
LX.trivial <- matrix.object(loading.X.trivial, "u2")
error.cor.X.trivial <- matrix(NA, 6, 6)
diag(error.cor.X.trivial) <- 0
TD.trivial <- sym.matrix.object(error.cor.X.trivial, "n1")
error.cor.Y.trivial <- matrix(NA, 2, 2)
diag(error.cor.Y.trivial) <- 0
TE.trivial <- sym.matrix.object(error.cor.Y.trivial, "n1")
TH.trivial <- matrix.object(matrix(NA, 6, 2), "n1")
SEM.Mis.Model <- misspecified.SEM.object(LX = LX.trivial, TE = TE.trivial, TD = TD.trivia
```

`model.object` *Create model object from model specification and be ready for data analysis.*

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` that provides model specification
`...`	Other values that will be explained specifically for each class

Value

`simModel` that will be used for data analysis

Details in ...

1. *Constraint:* simConstrint.c that save constraints specified by users. The default is no constraint.
2. *Program:* Desired analysis package

Author(s)

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

See Also

1. `simModel` for the result object
2. `simMatrixSet` for the target object containing model specification

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

`plot.cutoff` *Plot sampling distributions of fit indices*

Description

This function will plot sampling distributions of null hypothesis fit indices. The users may add cutoffs by specifying the `alpha` level.

Usage

```
plot.cutoff(object, ...)
```

Arguments

`object`	The object (`simResult` or `data.frame`) that contains values of fit indices in each distribution.
`...`	Other arguments specific to different types of object you pass in the function.

Value

NONE. Only plot the fit indices distributions.

Details in ...

1. `cutoff`: A priori cutoffs for fit indices, saved in a vector
2. `alpha`: A priori alpha level to find cutoffs of fit indices (do not specify when you have `cutoff`)
3. `reverse`: The default is to find critical point on the side that indicates worse fit (the right side of RMSEA or the left side of CFI). If specifying as TRUE, the directions are reversed.
4. `used.fit`: The name of fit indices that researchers wish to plot

Author(s)

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

See Also

1. `simResult` for result object that used in this function.
2. `find.cutoff` to find values of cutoffs based on null hypothesis sampling distributions only

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)
CFA.Model <- matrix.CFA.object(LY = LX, PS = PH, TE = TD)
SimData <- data.object(200, CFA.Model)
SimModel <- model.object(CFA.Model)
# We make the examples running only 50 replications to save time.
# In reality, more replications are needed.
Output <- result.object(SimData, SimModel, 50)
plot.cutoff(Output, 0.05, used.fit=c("RMSEA", "SRMR", "CFI", "TLI"))
```

---

plot.power *Plot sampling distributions of fit indices that visualize power*

Description

This function will plot sampling distributions of fit indices that visualize power in either a histogram or overlapping histograms.

Usage

```
plot.power(object.alt, object.null, ...)
```

Arguments

object.alt	The object (simResult or data.frame) that saves fit indices for alternative hypothesis
object.null	The object that represents null hypothesis. It can be vector of cutoffs (that might be calculated from find.cutoff or an object that save raw data of fit indices for null hypothesis (simResult or data.frame).
...	Other arguments specific to different types of object you pass in the function.

Value

NONE. Only plot the fit indices distributions.

Details in ...

1. alpha: A priori alpha level to find cutoffs of fit indices (do not specify when you have cutoff)
2. used.fit: The name of fit indices that researchers wish to plot

Author(s)

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

See Also

1. simResult for result object that used in this function.
2. find.cutoff to find values of cutoffs based on null hypothesis sampling distributions only

Examples

```
loading.null <- matrix(0, 6, 1)
loading.null[1:6, 1] <- NA
LX.NULL <- matrix.object(loading.null, 0.7)
PH.NULL <- sym.matrix.object(diag(1))
TD <- sym.matrix.object(diag(6))
CFA.Model.NULL <- matrix.CFA.object(LY = LX.NULL, PS = PH.NULL, TE = TD)
SimData.NULL <- data.object(500, CFA.Model.NULL)
SimModel <- model.object(CFA.Model.NULL)
# We make the examples running only 50 replications to save time.
# In reality, more replications are needed.
Output.NULL <- result.object(SimData.NULL, SimModel, 50)
Cut.NULL <- find.cutoff(Output.NULL, 0.95)

u79 <- runif.object(0.7, 0.9)
loading.alt <- matrix(0, 6, 2)
loading.alt[1:3, 1] <- NA
loading.alt[4:6, 2] <- NA
LX.ALT <- matrix.object(loading.alt, 0.7)
latent.cor.alt <- matrix(NA, 2, 2)
diag(latent.cor.alt) <- 1
PH.ALT <- sym.matrix.object(latent.cor.alt, "u79")
CFA.Model.ALT <- matrix.CFA.object(LY = LX.ALT, PS = PH.ALT, TE = TD)
SimData.ALT <- data.object(500, CFA.Model.ALT)
Output.ALT <- result.object(SimData.ALT, SimModel, 50)
find.power(Output.ALT, Cut.NULL)
Rule.of.thumb <- c(RMSEA=0.05, CFI=0.95, TLI=0.95, SRMR=0.06)
plot.power(Output.ALT, Output.NULL, alpha=0.05, used.fit=c("RMSEA", "CFI", "TLI", "SRMR")
```

---

result.object *Create result object.*

Description

This function will create result object by different ways. One way is to create data and analyze data multiple times by specifying simData and simModel and save it in the simResult.

Usage

```
result.object(simData, simModel, NRep, seed = 123321, silent = FALSE)
```

Arguments

simData	Data object used in data simulation.
simModel	Model object used in analyzing the simulated data.
NRep	Number of replications.
seed	Seed number
silent	TRUE if users do not wish to print number of replications during running the function.

Value

simResult that saves analysis result from simulate data.

Author(s)

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

See Also

1. simData for data model specification
2. simModel for analysis model specification
3. simResult for the type of resulting object

Examples

```
loading <- matrix(0, 6, 1)
loading[1:6, 1] <- NA
LX <- matrix.object(loading, 0.7)
PH <- sym.matrix.object(diag(1))
TD <- sym.matrix.object(diag(6))
CFA.Model <- matrix.CFA.object(LY = LX, PS = PH, TE = TD)
SimData <- data.object(500, CFA.Model)
SimModel <- model.object(CFA.Model)
# We make the examples running only 50 replications to save time.
# In reality, more replications are needed.
Output <- result.object(SimData, SimModel, 50)
#summary(Output)
```

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>)

See Also

`simDist` for other distribution objects.

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 (`Rnorm`) that save the specified parameters

Author(s)

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

See Also

1. Rnorm for the result object.
2. simDist for other distribution objects.

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, listed below.

Value

object	depends on particular object

Methods

signature(object = "Rnorm") No additional arguments. The function will random draw a number from normal distribution object.

signature(object = "Runif") No additional arguments. The function will random draw a number from uniform distribution object.

signature(object = "simData") The function will random data from data object. Users may add N argument to change sample size.

signature(object = "simMatrix") No additional arguments. The function will random parameters from matrix object.

signature(object = "simMatrixSet") No additional arguments. The function will random parameters from set of matrix objects and vector objects.

signature(object = "simMisspecifiedSet") No additional arguments. The function will random parameters from set of matrix objects and vector objects in model misspecification.

`signature(object = "simModel")` The function will run an analysis specified in the `simModel` object. One additional required argument is the data (put it as the second argument)

`signature(object = "simVector")` No additional arguments. The function will random parameters from vector object.

`signature(object = "symMatrix")` No additional arguments. The function will random parameters from symmetric matrix object.

Author(s)

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

See Also

This is the list of classes that can use `run` method.

1. `Rnorm`
2. `Runif`
3. `simMatrix`
4. `symMatrix`
5. `simVector`
6. `simMatrixSet`
7. `simData`
8. `simModel`
9. `simMisspecifiedSet`

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>)

See Also

`simDist` for other distribution objects.

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 (`Runif`) that save the specified parameters

Author(s)

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

See Also

1. `Runif` for the result object.
2. `simDist` for other distribution objects.

Examples

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

---

simConstraint-class

Class "simConstraint"

Description

Set of specified equality constraints

Details

The Equality slot contains list of equality constraint. Each element in the list is an individual equality constraint saved in a matrix. Each row represents each element. If the matrix has two columns, the first column indicates row of the element and the second column indicates column of the element. If the matrix has three columns, the first column is the group of matrix. The rest is row and column. Row name represents the matrix that the element is in. The definition of row name can be seen in matrix.CFA.object, matrix.Path.object, or matrix.SEM.object, depending on analysis model you specify.

Objects from the Class

Objects can be created by constraint.object. Also, it can be called of the form new("simConstraint", ...).

Slots

Equality: List of equality constraint. See the Details section for the description of each equality constraint.

Tag: Analysis model (CFA, SEM, Path)

Methods

summary Summarize all attributes of this object

Author(s)

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

See Also

1. constraint.object for the constructor of this class
2. data.object for a potential use of this object to create data
3. model.object for a potential use of this object to run an analysis

Examples

```
showClass("simConstraint")
constraint1 <- matrix(1, 3, 2)
constraint1[,1] <- 1:3
rownames(constraint1) <- rep("LY", 3)
constraint2 <- matrix(2, 3, 2)
constraint2[,1] <- 4:6
rownames(constraint2) <- rep("LY", 3)
constraint3 <- matrix(3, 2, 2)
constraint3[,1] <- 7:8
rownames(constraint3) <- rep("LY", 2)
equal.loading <- constraint.object(constraint1, constraint2, constraint3, Tag="SEM")
summary(equal.loading)
```

---

simData-class *Class* "simData"

Description

This class will save information for data simulation and can create data by run function.

Objects from the Class

Objects can be created by `data.object`. Also, it can be called by `new("simData", ...)`.

Slots

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

`N`: Sample size

`Parameters`: Model specification that used in data generation. It must be in `simMatrixSet` class.

`Misspecified`: Model misspecification that used in data generation. It must be in `simMisspecifiedSet` class.

`Constraint`: Equality constraints in data generation. It must be in `simConstraint` class.

`Constrain.Parameters.Only`: `TRUE` if users wish to constrain parameters before adding misspecification. `FALSE` if users wish to constrain parameters after adding misspecification.

`Misfit.bound`: Upper bound of population RMSEA that users wish their model misspecification to be

`Maximum.random`: The maximum number of random drawn parameters and misspecification model until all parameters in the model are eligible (no negative error variance, standardized coefficients over 1).

Methods

run To create data from this class. `N` is the additional argument that users may change the sample size when creating data.

summary Summarize all attributes in the data object.

Author(s)

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

See Also

1. linkS4class{simMatrixSet} for how to specify data generation model.
2. linkS4class{simMisspecifiedSet} for how to specify misspecification in this data generation model.
3. linkS4class{simConstraint} for how to set equality constraints for data generation.
4. link{result.object} for the use of this class to run Monte Carlo simulation.

Examples

```
showClass("simData")
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)
CFA.Model <- matrix.CFA.object(LY = LX, PS = PH, TE = TD)
SimData <- data.object(200, CFA.Model)
summary(SimData)
run(SimData)
```

---

simDist-class *Class "simDist"*

Description

All distribution objects. (Virtual Class)

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>)

See Also

List of all distribution objects.

1. Rnorm
2. Runif

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

1. symMatrix for symmetric random parameter matrix
2. 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)
```

---

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.CFA.object`, `matrix.Path.object`, or `matrix.SEM.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

run Create a sample of parameters in this object. In other words, draw a sample from all random parameters which is represented in `simDist`.

summary Get the summary of model specification

Author(s)

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

See Also

1. Create an object this class by CFA, Path Analysis, or SEM model by `matrix.CFA.object`, `matrix.Path.object`, or `matrix.SEM.object`, respectively.
2. See how to specify model misspecification by `simMisspecifiedSet`.

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)
```

`simMisspecifiedSet-class`
Class `"simMisspecifiedSet"`

Description

Misspecification model added on true model specifiction. This class contains `simVector`, `simMatrix`, and `symMatrix` specifying misspecification.

Objects from the Class

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

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

Extends

Class "simMatrixSet", directly.

Methods

run Create a sample of parameters in this object. In other words, draw a sample from all random parameters which is represented in simDist.

Author(s)

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

See Also

1. Create an object this class by CFA, Path Analysis, or SEM model by misspecified.CFA.object, misspecified.Path.object, or misspecified.SEM.object, respectively.
2. See how to specify true model by simMatrixSet.

Examples

```
showClass("simMisspecifiedSet")
n01 <- rnorm.object(0, 0.1)
error.cor.Mis <- matrix(NA, 6, 6)
diag(error.cor.Mis) <- 1
TD.Mis <- sym.matrix.object(error.cor.Mis, "n01")
CFA.Model.Mis <- misspecified.CFA.object(TD=TD.Mis)
```

---

simModel-class *Class* "simModel"

Description

This class will save information for analysis model and be ready for data analysis.

Objects from the Class

Objects can be created by model.object. It can also be called by new("simModel", ...).

Slots

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

Parameters: Set of all free parameters and values of fixed parameters in the model.

Starting.Values: All starting values of free parameters

Constraint: Equality constraints in simConstraint class

Program: Packages used in data analysis, either lavaan or OpenMx. The default is lavaan

Methods

run signature(object = "simModel"): ...

Author(s)

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

See Also

1. model.object for the constructor of this class.
2. simConstraint for specifying equality constraints.

Examples

```
showClass("simModel")
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)
CFA.Model <- matrix.CFA.object(LX = LX, PH = PH, TD = TD)
SimModel <- model.object(CFA.Model)
#summary(SimModel)
```

---

simResult-class *Class* "simResult"

Description

This class will save data analysis results from multiple replications and ready to find some useful statistics, such as fit indices cutoffs or power.

Objects from the Class

Objects can be created by result.object. It can also be called from the form new("simResult", ...).

Slots

Tag: Analysis model type (CFA, Path, or SEM)

Data: Data object that save data generation model. It must be in simData class.

Model: Model object that save analysis model. It must be in simModel class.

Replication: Number of replications have been created and run simulated data.

Output: Fit Indices values from each replication

Convergence: Number of convergence replications

Seed: Seed number.

Methods

1. find.cutoff to find cutoff of fit indices based on a priori alpha level.
2. find.power to find power of rejection when the result object is the alternative hypothesis and users specify cutoffs of the fit indices.
3. plot.cutoff to plot null hypothesis sampling distributions of fit indices with an option to draw fit indices cutoffs by specifying a priori alpha level.
4. plot.power to plot alternative hypothesis (and null hypothesis) with a priori cutoffs or alpha level.

Author(s)

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

See Also

1. simData for data generation model.
2. simModel for analysis model
3. result.object for the constructor of this class

Examples

```
showClass("simResult")
loading <- matrix(0, 6, 1)
loading[1:6, 1] <- NA
LX <- matrix.object(loading, 0.7)
PH <- sym.matrix.object(diag(1))
TD <- sym.matrix.object(diag(6))
CFA.Model <- matrix.CFA.object(LY = LX, PS = PH, TE = TD)
SimData <- data.object(500, CFA.Model)
SimModel <- model.object(CFA.Model)
# We make the examples running only 50 replications to save time.
# In reality, more replications are needed.
Output <- result.object(SimData, SimModel, 50)
#summary(Output)
find.cutoff(Output, 0.95)
```

---

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)
```

---

summary-methods *Methods for function* `summary` *in package* **simsem**

Description

Provide a summary of all attributes in an object

Usage

```
summary(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

summary.short for a shorter version of summary.

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(LX)
```

---

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

This is the list of classes that can use `run` method.

1. `simMatrix`
2. `simVector`

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

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

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)
```

---

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

1. See simVector for the resulting object.
2. See matrix.object for creating matrix object.
3. See sym.matrix.object for creating 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