Package 'simsem'

November 16, 2011

Type Package

Title SIMulated Structural Equation Modeling data.

Version 0.0-2

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:	0.0.2
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` *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(target, param, pos, numAsFixed)
```

Arguments

`target`	Target `SimMatrix`, `SymMatrix`, or `SimVector` that you would like to adjust.
`param`	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.
`pos`	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.
`numAsFixed`	This argument is used when the `VirtualDist` 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 `pos` 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 <- simMatrix(loading, 0.7)
summary(LX)
run(LX)

u34 <- simUnif(0.3, 0.4)
LX <- adjust(LX, "u34", c(2, 1))
summary(LX)
run(LX)

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

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

n01 <- simNorm(0, 1)
AL <- adjust(AL, "n01", 2)
run(AL)
summary(AL)
```

`getCutoff` *Find cutoff given a priori alpha level*

Description

Extract fit indices information from the `SimResult` and getCutoff of fit indices given a priori alpha level

Usage

```
getCutoff(object, alpha, revDirec=FALSE, usedFit=NULL)
```

Arguments

`object`	`SimResult` that saves the analysis results from multiple replications
`alpha`	A priori alpha level
`revDirec`	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.
`usedFit`	Vector of names of fit indices that researchers wish to getCutoffs from. The default is to getCutoffs 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 simResult

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 <- simMatrix(loading, loadingValues)
latent.cor <- matrix(NA, 2, 2)
diag(latent.cor) <- 1
PH <- symMatrix(latent.cor, 0.5)
error.cor <- matrix(0, 6, 6)
diag(error.cor) <- 1
TD <- symMatrix(error.cor)
CFA.Model <- simSetCFA(LY = LX, PS = PH, TE = TD)
SimData <- simData(200, CFA.Model)
SimModel <- simModel(CFA.Model)
# We make the examples running only 50 replications to save time.
# In reality, more replications are needed.
Output <- simResult(SimData, SimModel, 50)
getCutoff(Output, 0.05)
```

---

getPower *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 getCutoff of the null model.

Usage

```
getPower(altObject, cutoff, revDirec = FALSE, usedFit=NULL)
```

Arguments

altObject	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 getCutoff function.

revDirec — 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.

usedFit — Vector of names of fit indices that researchers wish to getCutoffs from. The default is to getCutoffs 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. getCutoff to find the cutoffs from null model.
2. SimResult to see how to create simResult

Examples

```
loading.null <- matrix(0, 6, 1)
loading.null[1:6, 1] <- NA
LX.NULL <- simMatrix(loading.null, 0.7)
PH.NULL <- symMatrix(diag(1))
TD <- symMatrix(diag(6))
CFA.Model.NULL <- simSetCFA(LY = LX.NULL, PS = PH.NULL, TE = TD)
SimData.NULL <- simData(500, CFA.Model.NULL)
SimModel <- simModel(CFA.Model.NULL)
# We make the examples running only 50 replications to save time.
# In reality, more replications are needed.
Output.NULL <- simResult(SimData.NULL, SimModel, 50)
Cut.NULL <- getCutoff(Output.NULL, 0.95)

u79 <- simUnif(0.7, 0.9)
loading.alt <- matrix(0, 6, 2)
loading.alt[1:3, 1] <- NA
loading.alt[4:6, 2] <- NA
LX.ALT <- simMatrix(loading.alt, 0.7)
latent.cor.alt <- matrix(NA, 2, 2)
diag(latent.cor.alt) <- 1
PH.ALT <- symMatrix(latent.cor.alt, "u79")
CFA.Model.ALT <- simSetCFA(LY = LX.ALT, PS = PH.ALT, TE = TD)
SimData.ALT <- simData(500, CFA.Model.ALT)
Output.ALT <- simResult(SimData.ALT, SimModel, 50)
getPower(Output.ALT, Cut.NULL)
Rule.of.thumb <- c(RMSEA=0.05, CFI=0.95, TLI=0.95, SRMR=0.06)
getPower(Output.ALT, Rule.of.thumb, usedFit=c("RMSEA", "CFI", "TLI", "SRMR"))
```

loadingFromAlpha *Find standardized factor loading from coefficient alpha*

Description

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

Usage

```
loadingFromAlpha(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

```
loadingFromAlpha(0.8, 4)
```

plotCutoff *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

```
plotCutoff(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 getCutoffs of fit indices (do not specify when you have `cutoff`)
3. `revDirec`: 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. `usedFit`: 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 simResult that used in this function.
2. `getCutoff` 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 <- simMatrix(loading, loadingValues)
latent.cor <- matrix(NA, 2, 2)
diag(latent.cor) <- 1
PH <- symMatrix(latent.cor, 0.5)
error.cor <- matrix(0, 6, 6)
diag(error.cor) <- 1
TD <- symMatrix(error.cor)
CFA.Model <- simSetCFA(LY = LX, PS = PH, TE = TD)
SimData <- simData(200, CFA.Model)
SimModel <- simModel(CFA.Model)
# We make the examples running only 50 replications to save time.
# In reality, more replications are needed.
Output <- simResult(SimData, SimModel, 50)
plotCutoff(Output, 0.05, usedFit=c("RMSEA", "SRMR", "CFI", "TLI"))
```

`plotPower` *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

```
plotPower(altObject, nullObject, ...)
```

Arguments

altObject	The object (SimResult or data.frame) that saves fit indices for alternative hypothesis
nullObject	The object that represents null hypothesis. It can be vector of cutoffs (that might be calculated from getCutoff 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 getCutoffs of fit indices (do not specify when you have cutoff)
2. usedFit: 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 simResult that used in this function.
2. getCutoff 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 <- simMatrix(loading.null, 0.7)
PH.NULL <- symMatrix(diag(1))
TD <- symMatrix(diag(6))
CFA.Model.NULL <- simSetCFA(LY = LX.NULL, PS = PH.NULL, TE = TD)
SimData.NULL <- simData(500, CFA.Model.NULL)
SimModel <- simModel(CFA.Model.NULL)
# We make the examples running only 50 replications to save time.
# In reality, more replications are needed.
Output.NULL <- simResult(SimData.NULL, SimModel, 50)
Cut.NULL <- getCutoff(Output.NULL, 0.95)

u79 <- simUnif(0.7, 0.9)
loading.alt <- matrix(0, 6, 2)
loading.alt[1:3, 1] <- NA
loading.alt[4:6, 2] <- NA
LX.ALT <- simMatrix(loading.alt, 0.7)
latent.cor.alt <- matrix(NA, 2, 2)
diag(latent.cor.alt) <- 1
PH.ALT <- symMatrix(latent.cor.alt, "u79")
CFA.Model.ALT <- simSetCFA(LY = LX.ALT, PS = PH.ALT, TE = TD)
SimData.ALT <- simData(500, CFA.Model.ALT)
Output.ALT <- simResult(SimData.ALT, SimModel, 50)
getPower(Output.ALT, Cut.NULL)
```

```
Rule.of.thumb <- c(RMSEA=0.05, CFI=0.95, TLI=0.95, SRMR=0.06)
plotPower(Output.ALT, Output.NULL, alpha=0.05, usedFit=c("RMSEA", "CFI", "TLI", "SRMR"))
```

`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 = "SimNorm")` No additional arguments. The function will random draw a number from normal distribution object.

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

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

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

`signature(object = "SimSet")` No additional arguments. The function will random parameters from set of simMatrixs and simVectors.

`signature(object = "SimMisspec")` No additional arguments. The function will random parameters from set of simMatrixs and simVectors 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 simVector.

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

Author(s)

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

See Also

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

1. SimNorm
2. SimUnif
3. SimMatrix
4. SymMatrix
5. SimVector
6. SimSet
7. SimData
8. SimModel
9. SimMisspec

Examples

```
n02 <- simNorm(0, 0.2)
run(n02)
```

`simData` *Data object*

Description

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

Usage

```
simData(n, param, misspec = new("NullSimMisspec"), equalCon = new("NullSimEqualC
```

Arguments

`n`	Desired sample size
`param`	Model specification matrices that are created by `simSetCFA`, `simSetPath`, or `simSetSEM`.
`misspec`	Model *misspecification* matrices that are created by `simMisspecCFA`, `simMisspecPath`, or `simMisspecSEM`.
`equalCon`	Equality constraints that are created by `simEqualCon`. This will specify equality econstraints of parameters in data generation process.
`conBeforeMis`	`TRUE` if users wish to constrain parameters before adding misspecification. `FALSE` if users wish to constrain parameters after adding misspecification.
`misfitBound`	Upper bound of population root mean squared error of approximation (RMSEA; Browne & Cudeck, 1992) that users wish their model misspecification to be
`maxDraw`	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

- `simSetCFA` to see CFA model specification
- `simSetPath` to see Path analysis model specification
- `simSetSEM` to see SEM model specification
- `simMisspecCFA` for specifying misspecification in CFA model
- `simMisspecPath` for specifying misspecification in Path analysis model
- `simMisspecSEM` for specifying misspecification in SEM model
- `simEqualCon` 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 <- simMatrix(loading, loadingValues)
latent.cor <- matrix(NA, 2, 2)
diag(latent.cor) <- 1
PH <- symMatrix(latent.cor, 0.5)
error.cor <- matrix(0, 6, 6)
diag(error.cor) <- 1
TD <- symMatrix(error.cor)
CFA.Model <- simSetCFA(LY = LX, PS = PH, TE = TD)
SimData <- simData(200, CFA.Model)
summary(SimData)
run(SimData)

# With Misspecification Model
n01 <- simNorm(0, 0.1)
error.cor.Mis <- matrix(NA, 6, 6)
```

```
diag(error.cor.Mis) <- 1
TD.Mis <- symMatrix(error.cor.Mis, "n01")
CFA.Model.Mis <- simMisspecCFA(TD=TD.Mis)
SimData <- simData(200, CFA.Model, misspec=CFA.Model.Mis)
summary(SimData)
run(SimData)
```

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 simData. Also, it can be called by new("SimData", ...).

Slots

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

n: Sample size

param: Model specification that used in data generation. It must be in SimSet class.

misspec: Model misspecification that used in data generation. It must be in SimMisspec class.

equalCon: Equality constraints in data generation. It must be in SimEqualCon class.

conBeforeMis: TRUE if users wish to constrain parameters before adding misspecification. FALSE if users wish to constrain parameters after adding misspecification.

misfitBound: Upper bound of population RMSEA that users wish their model misspecification to be

maxDraw: 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. dataOnly is default to be TRUE. If FALSE, the resulting object in SimDataOut can be used to provide details of things used in create the data.

summary Summarize all attributes in the simData.

Author(s)

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

See Also

- linkS4class{SimSet} for how to specify data generation model.
- linkS4class{SimMisspec} for how to specify misspecification in this data generation model.
- linkS4class{SimEqualCon} for how to set equality constraints for data generation.
- link{simResult} for the use of this class to run Monte Carlo simulation.
- linkS4class{SimModelOut} for the output type after the run function with dataOnly=TRUE.

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 <- simMatrix(loading, loadingValues)
latent.cor <- matrix(NA, 2, 2)
diag(latent.cor) <- 1
PH <- symMatrix(latent.cor, 0.5)
error.cor <- matrix(0, 6, 6)
diag(error.cor) <- 1
TD <- symMatrix(error.cor)
CFA.Model <- simSetCFA(LY = LX, PS = PH, TE = TD)
SimData <- simData(200, CFA.Model)
summary(SimData)
run(SimData)
```

SimDataOut-class *Class* "SimDataOut"

Description

This class will provide the simulated dataset and population behind the generated dataset.

Objects from the Class

Objects can be created by run on the SimData with dataOnly=FASE. It can also be called from the form new("SimDataOut", ...).

Slots

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

data: The simulated data

param: Model specification that used in data generation. It must be in SimSet class.

paramOut: Parameter values underlying the simulated data.

misspecOut: Model misspecification underlying the simulated data

equalCon: Equality constraints in data generation. It must be in SimEqualCon class.

Methods

- summaryTo summarize the object

Author(s)

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

See Also

- SimData The object used as data template for simulated data.
- SimModel The object used as analysis

Examples

```
showClass("SimDataOut")
loading <- matrix(0, 6, 1)
loading[1:6, 1] <- NA
LX <- simMatrix(loading, 0.7)
PH <- symMatrix(diag(1))
TD <- symMatrix(diag(6))
CFA.Model <- simSetCFA(LY = LX, PS = PH, TE = TD)
SimData <- simData(500, CFA.Model)
SimModel <- simModel(CFA.Model)
Data <- run(SimData, dataOnly=FALSE)
Result <- run(SimModel, Data)
summary(Data)
```

---

simEqualCon *Equality Constraint Object*

Description

This function will be used to specify equality constraints.

Usage

```
simEqualCon(..., modelType)
```

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.
modelType	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 simSetCFA function. If users specify Path analysis with or without exogenous variables, the specification is shown in simSetPath function. If users specify SEM model with or without exogenous variables, the specification is shown in simSetSEM function. However, basically, the names of matrices you put in these function are also eligible for this function as well.

Value

Object in SimEqualCon 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

- simSetCFA to see model specification in CFA model
- simSetPath to see model specification in Path analysis model
- simSetSEM to see model specification in SEM model
- SimEqualCon for the simResult

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 <- simEqualCon(constraint, modelType="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 <- simEqualCon(group.con, modelType="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 <- simEqualCon(constraint1, constraint2, constraint3, modelType="SEM")
summary(equal.loading2)
```

SimEqualCon-class *Class* "SimEqualCon"

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 simSetCFA, simSetPath, or simSetSEM, depending on analysis model you specify.

Objects from the Class

Objects can be created by simEqualCon. Also, it can be called of the form new("SimEqualCon", ...).

Slots

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

modelType: 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

- simEqualCon for the constructor of this class
- simData for a potential use of this object to create data
- simModel for a potential use of this object to run an analysis

Examples

```
showClass("SimEqualCon")
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 <- simEqualCon(constraint1, constraint2, constraint3, modelType="SEM")
summary(equal.loading)
```

---

simMatrix *Create simMatrix 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

```
simMatrix(free, param = NULL)
```

Arguments

`free`	Matrix of free parameters. Use NA to specify free parameters. Use number as fixed value (including zero)
`param`	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 `VirtualDist`.

Value

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

Author(s)

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

See Also

- See `VirtualDist` for the resulting object.
- See `symMatrix` for creating symmetric simMatrix.
- See `simVector` for simVector.

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 <- simMatrix(loading, loadingValues)
summary(LX)
run(LX)

n65 <- simNorm(0.6, 0.05)
LY <- simMatrix(loading, "n65")
summary(LY)
run(LY)
```

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 "simMatrix" function. Objects can be also created by calls of the form new("SimMatrix", ...).

Slots

free: 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.

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

Methods

adjust 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.

summaryShort 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
- 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 <- simMatrix(loading, loadingValues)
summary(LX)
run(LX)

n65 <- simNorm(0.6, 0.05)
LY <- simMatrix(loading, "n65")
summary(LY)
run(LY)

u34 <- simUnif(0.3, 0.4)
LY <- adjust(LY, "u34", c(2, 1))
summary(LY)
run(LY)
summaryShort(LY)
```

SimMisspec-class *Class* "SimMisspec"

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 simMisspecCFA, simMisspecPath, or simMisspecSEM, for CFA, Path analysis, or SEM model, respectively. Objects can be also created by calls of the form new("SimMisspec", ...).

Slots

modelType: 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 "SimSet", 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 VirtualDist.

Author(s)

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

See Also

- Create an object this class by CFA, Path Analysis, or SEM model by simMisspecCFA, simMisspecPath, or simMisspecSEM, respectively.
- See how to specify true model by SimSet.

Examples

```
showClass("SimMisspec")
n01 <- simNorm(0, 0.1)
error.cor.Mis <- matrix(NA, 6, 6)
diag(error.cor.Mis) <- 1
TD.Mis <- symMatrix(error.cor.Mis, "n01")
CFA.Model.Mis <- simMisspecCFA(TD=TD.Mis)
```

simMisspecCFA *Set of model misspecification for CFA model.*

Description

This function will define model misspecifiction from a defined model. This function is similar to simSetCFA 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

```
simMisspecCFA(...)
```

Arguments

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

Value

object in SimMisspec that saves model misspecification.

Author(s)

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

See Also

- simSetCFA for matrix definition and how to specify CFA model
- SimMisspec for the simResult
- simMisspecPath for misspecification model in Path analysis and simMisspecSEM for misspecification model in SEM.

Examples

```
n01 <- simNorm(0, 0.1)
error.cor.Mis <- matrix(NA, 6, 6)
diag(error.cor.Mis) <- 1
TD.Mis <- symMatrix(error.cor.Mis, "n01")
CFA.Model.Mis <- simMisspecCFA(TD=TD.Mis)
```

simMisspecPath *Set of model misspecification for Path analysis model.*

Description

This function will define model misspecifiction from a defined model. This function is similar to simSetPath 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

```
simMisspecPath(..., exo = FALSE)
```

Arguments

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

Value

object in SimMisspec that saves model misspecification.

Author(s)

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

See Also

- simSetPath for matrix definition and how to specify Path analysis model
- SimMisspec for the simResult
- simMisspecCFA for misspecification model in CFA and simMisspecSEM for misspecification model in SEM.

Examples

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

simMisspecSEM *Set of model misspecification for SEM model.*

Description

This function will define model misspecifiction from a defined model. This function is similar to simSetSEM 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

```
simMisspecSEM(..., exo = FALSE)
```

Arguments

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

Value

object in SimMisspec that saves model misspecification.

Author(s)

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

See Also

- simSetSEM for matrix definition and how to specify SEM model
- SimMisspec for the simResult
- simMisspecCFA for misspecification model in CFA and simMisspecPath for misspecification model in Path analysis.

Examples

```
u2 <- simUnif(-0.2, 0.2)
n1 <- simNorm(0, 0.1)
loading.X.trivial <- matrix(NA, 6, 2)
loading.X.trivial[is.na(loading.X.trivial)] <- 0
LX.trivial <- simMatrix(loading.X.trivial, "u2")
error.cor.X.trivial <- matrix(NA, 6, 6)
diag(error.cor.X.trivial) <- 0
TD.trivial <- symMatrix(error.cor.X.trivial, "n1")
error.cor.Y.trivial <- matrix(NA, 2, 2)
diag(error.cor.Y.trivial) <- 0
TE.trivial <- symMatrix(error.cor.Y.trivial, "n1")
TH.trivial <- simMatrix(matrix(NA, 6, 2), "n1")
SEM.Mis.Model <- simMisspecSEM(LX = LX.trivial, TE = TE.trivial, TD = TD.trivial, TH = TH
```

simModel — *Create simModel from model specification and be ready for data analysis.*

Description

This function will take model specification from SimSet 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

```
simModel(object, ...)
```

Arguments

object	SimSet 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 ...

- *start:* SimRSet.c that saves all starting values in the model.
- *equalCon:* SimEqualCon.c that save constraints specified by users. The default is no constraint.
- *package:* Desired analysis package

Author(s)

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

See Also

- SimModel for the simResult
- SimSet 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 <- simMatrix(loading, loadingValues)
latent.cor <- matrix(NA, 2, 2)
diag(latent.cor) <- 1
PH <- symMatrix(latent.cor, 0.5)
error.cor <- matrix(0, 6, 6)
```

```
diag(error.cor) <- 1
TD <- symMatrix(error.cor)
CFA.Model <- simSetCFA(LX = LX, PH = PH, TD = TD)
SimModel <- simModel(CFA.Model)
```

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 simModel. It can also be called by new("SimModel", ...).

Slots

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

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

start: All starting values of free parameters

equalCon: Equality constraints in SimEqualCon class

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

Methods

run Analyze data. The first argument is the SimModel and the second argument is data saved in data.frame

summary To summarize the object

Author(s)

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

See Also

- simModel for the constructor of this class.
- SimEqualCon 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 <- simMatrix(loading, loadingValues)
latent.cor <- matrix(NA, 2, 2)
diag(latent.cor) <- 1
PH <- symMatrix(latent.cor, 0.5)
```

```
error.cor <- matrix(0, 6, 6)
diag(error.cor) <- 1
TD <- symMatrix(error.cor)
CFA.Model <- simSetCFA(LX = LX, PH = PH, TD = TD)
SimModel <- simModel(CFA.Model)
summary(SimModel)
```

SimModelOut-class *Class* "SimModelOut"

Description

This class will save the analysis results from a single analysis.

Objects from the Class

Objects can be created by run on the SimModel. It can also be called from the form new("SimModelOut", ...).

Slots

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

start: All starting values of free parameters

equalCon: Equality constraints in SimEqualCon class

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

coef: Parameter estimates saved in matrix arrangement

se: Standard errors of parameter saved in matrix arrangement

fit: Fit Indices values from each replication

converged: Number of convergence replications

paramValue: The parameter values behind the analyzed data.

Methods

- summaryTo summarize the object
- summaryParamTo summarize only parameter estimates, standard errors, and significance

Author(s)

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

See Also

- SimModel for analysis model

Examples

```
showClass("SimResult")
loading <- matrix(0, 6, 1)
loading[1:6, 1] <- NA
LX <- simMatrix(loading, 0.7)
PH <- symMatrix(diag(1))
TD <- symMatrix(diag(6))
CFA.Model <- simSetCFA(LY = LX, PS = PH, TE = TD)
SimData <- simData(500, CFA.Model)
SimModel <- simModel(CFA.Model)
Data <- run(SimData)
Result <- run(SimModel, Data)
summary(Result)
```

---

simNorm *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

```
simNorm(mean, sd)
```

Arguments

mean	Desired population mean
sd	Desired population standard deviation

Value

SimNorm	Random Normal Distribution object (SimNorm) that save the specified parameters

Author(s)

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

See Also

- SimNorm for the simResult.
- VirtualDist for other distribution objects.

Examples

```
n02 <- simNorm(0, 0.2)
run(n02)
```

SimNorm-class *Class "SimNorm"*

Description

Object that create a random number from normal distribution.

Objects from the Class

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

Slots

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

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

Extends

Class `"VirtualDist"`, directly.

Methods

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

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

Author(s)

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

See Also

`VirtualDist` for other distribution objects.

Examples

```
showClass("SimNorm")
n2 <- simNorm(0, 0.2)
run(n2)
summary(n2)
```

simResult *Create simResult.*

Description

This function will create simResult 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

```
simResult(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>); Patrick Miller (University of Kansas; <patr1ckm@ku.edu>)

See Also

- SimData for data model specification
- SimModel for analysis model specification
- SimResult for the type of resulting object

Examples

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

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 simResult. It can also be called from the form new("SimResult", ...).

Slots

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

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

coef: Parameter estimates from each replication

se: Standard errors of parameter estimates from each replication

fit: Fit Indices values from each replication

converged: Number of convergence replications

seed: Seed number.

paramValue: Population model underlying each simulated dataset.

Methods

- getCutoff to getCutoff of fit indices based on a priori alpha level.
- getPower to getPower of rejection when the simResult is the alternative hypothesis and users specify cutoffs of the fit indices.
- plotCutoff to plot null hypothesis sampling distributions of fit indices with an option to draw fit indices cutoffs by specifying a priori alpha level.
- plotPower to plot alternative hypothesis (and null hypothesis) with a priori cutoffs or alpha level.
- summary to summarize the result output
- summaryParam to summarize all parameter estimates

Author(s)

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

See Also

- SimData for data generation model.
- SimModel for analysis model
- simResult for the constructor of this class

Examples

```
showClass("SimResult")
loading <- matrix(0, 6, 1)
loading[1:6, 1] <- NA
LX <- simMatrix(loading, 0.7)
PH <- symMatrix(diag(1))
TD <- symMatrix(diag(6))
CFA.Model <- simSetCFA(LY = LX, PS = PH, TE = TD)
SimData <- simData(500, CFA.Model)
SimModel <- simModel(CFA.Model)
# We make the examples running only 50 replications to save time.
# In reality, more replications are needed.
Output <- simResult(SimData, SimModel, 50)
summary(Output)
getCutoff(Output, 0.05)
```

SimSet-class *Class* "SimSet"

Description

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

Objects from the Class

Object can be created by simSetCFA, simSetPath, or simSetSEM, for CFA, Path analysis, or SEM model, respectively. Objects can be also created by calls of the form new("SimSet", ...).

Slots

modelType: 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 VirtualDist.

summary Get the summary of model specification

Author(s)

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

See Also

- Create an object this class by CFA, Path Analysis, or SEM model by simSetCFA, simSetPath, or simSetSEM, respectively.
- See how to specify model misspecification by SimMisspec.

Examples

```
showClass("SimSet")

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 <- simMatrix(loading, loadingValues)
summary(LX)

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

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

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

simSetCFA *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

```
simSetCFA(...)
```

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 simSetSEM 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

SimSet object that represents the CFA object. This will be used for specifying data or simModels later.

Author(s)

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

See Also

- See class SimSet for simResult details.
- See SimMatrix, SymMatrix, or SimVector for input details.
- Use simSetPath to specify path analysis model and use simSetSEM 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 <- simMatrix(loading, loadingValues)
summary(LX)

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

error.cor <- matrix(0, 6, 6)
diag(error.cor) <- 1
TD <- symMatrix(error.cor)

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

---

simSetPath *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

```
simSetPath(..., 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

- 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.

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

- 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). 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

SimSet object that represents the path analysis simModel. This will be used for specifying data or simModels later.

Author(s)

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

See Also

- See class SimSet for simResult details.
- See SimMatrix, SymMatrix, or SimVector for input details.
- Use simSetCFA to specify CFA model and use simSetSEM to specify full structural equation modeling.

Examples

```
u35 <- simUnif(0.3, 0.5)
u57 <- simUnif(0.5, 0.7)
u1 <- simUnif(-0.1, 0.1)
n31 <- simNorm(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 <- simMatrix(path.BE, starting.BE)

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

Path.Model <- simSetPath(PS = PS, BE = BE)

u35 <- simUnif(0.3, 0.5)
u57 <- simUnif(0.5, 0.7)
u1 <- simUnif(-0.1, 0.1)
n31 <- simNorm(0.3, 0.1)

path.GA <- matrix(0, 2, 2)
path.GA[1, 1:2] <- NA
GA <- simMatrix(path.GA, "u35")

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

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

PS <- symMatrix(diag(2))

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

---

simSetSEM *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

```
simSetSEM(..., 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

- 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.

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

- 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

SimSet object that represents the SEM object. This will be used for specifying data or simModels later.

Author(s)

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

See Also

- See class SimSet for simResult details.
- See SimMatrix, SymMatrix, or SimVector for input details.
- Use simSetCFA to specify CFA model and use simSetPath to specify path analysis model.

Examples

```
u68 <- simUnif(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 <- simMatrix(loading, loading.start)

TE <- symMatrix(diag(8))

factor.cor <- diag(3)
factor.cor[1, 2] <- factor.cor[2, 1] <- NA
PS <- symMatrix(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 <- simMatrix(path, path.start)

SEM.model <- simSetSEM(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 <- simMatrix(loading.X, 0.7)

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

TD <- symMatrix(diag(6))

TE <- symMatrix(diag(2))

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

PS <- symMatrix(as.matrix(1))

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

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

SEM.Exo.model <- simSetSEM(GA=GA, BE=BE, LX=LX, LY=LY, PH=PH, PS=PS, TD=TD, TE=TE, exo=TR
```

`simUnif` *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

```
simUnif(min, max)
```

Arguments

min	Lower bound of the distribution
max	Upper bound of the distribution

Value

SimUnif	Random Uniform Distribution object (SimUnif) that save the specified parameters

Author(s)

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

See Also

- SimUnif for the simResult.
- VirtualDist for other distribution objects.

Examples

```
u1 <- simUnif(-0.1, 0.1)
run(u1)
```

SimUnif-class *Class "SimUnif"*

Description

Object that create a random number from uniform distribution.

Objects from the Class

The object should be created by simUnif function. Objects can be created by calls of the form new("SimUnif", ...).

Slots

min: Lower bound parameter

max: Upper bound parameter

Extends

Class "VirtualDist", directly.

Methods

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

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

Author(s)

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

See Also

VirtualDist for other distribution objects.

Examples

```
showClass("SimUnif")
u1 <- simUnif(-0.1, 0.1)
run(u1)
summary(u1)
```

---

simVector *Create simVector 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

```
simVector(free, param = NULL)
```

Arguments

free	Vector of free parameters. Use NA to specify free parameters. Use number as fixed value (including zero).
param	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 VirtualDist.

Value

SimVector object that will be used for model specification later.

Author(s)

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

See Also

- See SimVector for the resulting object.
- See simMatrix for creating simMatrix.
- See symMatrix for creating symmetric simMatrix.

Examples

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

n02 <- simNorm(0, 0.2)
factor.start <- rep("n02", 4)
KA <- simVector(factor.mean, factor.start)
```

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 simVector function. Objects can be created by calls of the form new("SimVector", ...).

Slots

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

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

Methods

adjust 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.

summaryShort 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 <- simVector(factor.mean, factor.mean.starting)
run(AL)
summary(AL)
summaryShort(AL)

n01 <- simNorm(0, 1)
AL <- adjust(AL, "n01", 2)
run(AL)
summary(AL)
```

---

summaryParam *Provide summary of parameter estimates and standard error across replications*

Description

This function will provide averages of parameter estimates, standard deviations of parameter estimates, averages of standard errors, and power of rejection with a priori alpha level for the null hypothesis of parameters equal 0.

Usage

```
summaryParam(object, ...)
```

Arguments

object	SimResult object being described
...	any additional arguments, such as for the function with result object, detail argument is available. If TRUE, it provides relative bias, standardized bias, and relative bias in standard errors.

Value

A data frame that provides the statistics described above from all parameters. For using with linkS4class{SimModelOut}, each column means

- EstimateParameter Estimates
- SEStandard Error of the Parameter Estimates
- zWald Statistic
- p*p* value based on the Wald Statistic
- ParamParameter Value underlying the analyzed data

- BiasBias in Parameter Estimates
- CoverageWhether (1-alpha)% confidence interval covers the parameter estimates

For using with `linkS4class{SimResult}`, each column means

- Estimate.AverageAverage of parameter estimates across all replications
- Estimate.SDStandard Deviation of parameter estimates across all replications
- Average.SEAverage of standard errors across all replications
- Power (Not equal 0)Proportion of significant replications when testing whether the parameters are different from zero
- Average.ParamParameter values or average values of parameters if random parameters are specified
- SD.ParamStandard Deviations of parameters. Appeared only when random parameters are specified.
- Average.BiasThe difference between parameter estimates and parameter underlying data
- SD.BiasStandard Deviations of bias across all replications. Appeared only when random parameters are specified. This value is the expected value of average standard error when random parameter are specified.
- CoverageThe percentage of (1-alpha)% confidence interval covers parameters underlying the data.
- Rel.BiasRelative Bias, which is (`Estimate.Average - Average.Param`)/`Average.Param`. Hoogland and Boomsma (1998) proposed that the cutoff of .05 may be used for acceptable relative bias. This option will be available when `detail=TRUE`. This value will not be available when parameter values are very close to 0.
- Std.BiasStandardized Bias, which is (`Estimate.Average - Average.Param`)/`Estimate.SD` for fixed parameters and (`Estimate.Average - Average.Param`)/`SD.Bias` for random parameters. Collins, Schafer, and Kam (2001) recommended that biases will be only noticeable when standardized bias is greater than 0.4 in magnitude. This option will be available when `detail=TRUE`
- Rel.SE.BiasRelative Bias in standard error, which is (`Average.SE - Estimate.SD`)/`Estimate.SD` for fixed parameters and (`Average.SE - SD.Bias`)/`SD.Bias` for random parameters. Hoogland and Boomsma (1998) proposed that 0.10 is the acceptable level. This option will be available when `detail=TRUE`

Author(s)

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

References

- Collins, L. M., Schafer, J. L., & Kam, C. M. (2001). A comparison of inclusive and restrictive strategies in modern missing data procedures. *Psychological Methods, 6(4)*, 330.
- Hoogland, J. J., & Boomsma, A. (1998). Robustness studies in covariance structure modeling. *Sociological Methods & Research, 26(3)*, 329.

See Also

`SimResult` for the object input

Examples

```
showClass("SimResult")
loading <- matrix(0, 6, 1)
loading[1:6, 1] <- NA
LX <- simMatrix(loading, 0.7)
PH <- symMatrix(diag(1))
TD <- symMatrix(diag(6))
CFA.Model <- simSetCFA(LY = LX, PS = PH, TE = TD)
SimData <- simData(500, CFA.Model)
SimModel <- simModel(CFA.Model)
# We make the examples running only 50 replications to save time.
# In reality, more replications are needed.
Output <- simResult(SimData, SimModel, 50)
summaryParam(Output)
summaryParam(Output, detail=TRUE)
```

---

summaryShort *Provide short summary of an object.*

Description

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

Usage

```
summaryShort(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.

- SimMatrix
- SimVector

Examples

```
u89 <- simUnif(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 <- simMatrix(loading, "u89")
summaryShort(LX)
```

`symMatrix` *Create symmetric simMatrix 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

```
symMatrix(free, param = NULL)
```

Arguments

`free`	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.
`param`	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 `VirtualDist`.

Value

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

Author(s)

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

See Also

See `VirtualDist` for the resulting object. See `simMatrix` for creating simMatrix and `simVector` for simVector.

Examples

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

u46 <- simUnif(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 <- symMatrix(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 "symMatrix" function. Objects can be also created by calls of the form `new("SymMatrix", ...)`.

Slots

`free`**:** 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.

`param`**:** 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 `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 <- symMatrix(latent.cor, 0.5)

u46 <- simUnif(0.4, 0.6)
PH <- adjust(PH, "u46", c(3,2))
summary(PH)
summaryShort(PH)
run(PH)
```

VirtualDist-class *Class "VirtualDist"*

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 "VirtualDist" in the signature.

Author(s)

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

See Also

List of all distribution objects.

- SimNorm
- SimUnif

Examples

```
showClass("VirtualDist")
```

Index