Package 'semTools'

October 26, 2012

Type Package

Title Useful tools for structural equation modeling.

Version 0.2-8

Date 2012-10-26

Author Sunthud Pornprasertmanit <psunthud@ku.edu>, Patrick Miller <patr1ckm@ku.edu>, Alex Schoemann <schoemann@ku.edu>, Yves Rosseel <Yves.Rosseel@UGent.be>

Maintainer Sunthud Pornprasertmanit <psunthud@ku.edu>

Depends R(>= 2.14), MASS, lavaan, methods

Suggests parallel, Amelia, mice, foreign

Description This package provide useful tools for structural equation modeling analysis.

License GPL (>= 2)

LazyLoad yes

LazyData yes

URL https://github.com/simsem/semTools/wiki

R topics documented:

`auxiliary` *Analyzing data with full-information maximum likelihood with auxiliary variables*

Description

Analyzing data with full-information maximum likelihood with auxiliary variables. The techniques used to account for auxiliary variables are both extra-dependent-variables and saturated-correlates approaches (Enders, 2008). The extra-dependent-variables approach is used for exogenous varibles in the model (see `fixed.x` attribute when fitting the `lavaan` model). For other variables, the saturated-correlates approach is used. WARNINGS: this function does not work for all models. See the examples below.

Usage

```
auxiliary(object, aux, ...)
```

Arguments

`object`	The `lavaan` object or the parameter table
`aux`	The list of auxiliary variable
`...`	The additional arguments in the `lavaan` function.

Value

The `lavaanStar` object which contains the original `lavaan` object and the additional values of the null model, which need to be adjusted to account for auxiliary variables.

Author(s)

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

References

Enders, C. K. (2008). A note of the use of missing auxiliary variables in full information maximum likelihood-based structural equation models. *Structural Equation Modeling, 15*, 434-448.

See Also

lavaanStar

Examples

```
# Example of confirmatory factor analysis

HS.model <- ' visual  =~ x1 + x2 + x3
              textual =~ x4 + x5 + x6
              speed   =~ x7 + x8 + x9 '

dat <- data.frame(HolzingerSwineford1939, z=rnorm(nrow(HolzingerSwineford1939), 0, 1))

fit <- cfa(HS.model, data=dat) #, group="sex", meanstructure=TRUE)
fitaux <- auxiliary(fit, aux="z", data=dat)

# Example of multiple groups confirmatory factor analysis

fitgroup <- cfa(HS.model, data=dat, group="school")
fitgroupaux <- auxiliary(fitgroup, aux="z", data=dat, group="school")

# Example of path analysis

mod <- ' x5 ~ x4
x4 ~ x3
x3 ~ x1 + x2'

fitpath <- sem(mod, data=dat)
fitpathaux <- auxiliary(fitpath, aux="z", data=dat)

# Example of full structural equation modeling

dat2 <- data.frame(PoliticalDemocracy, z=rnorm(nrow(PoliticalDemocracy), 0, 1))
model <- '
     ind60 =~ x1 + x2 + x3
     dem60 =~ y1 + a*y2 + b*y3 + c*y4
     dem65 =~ y5 + a*y6 + b*y7 + c*y8

    dem60 ~ ind60
    dem65 ~ ind60 + dem60

    y1 ~~ y5
    y2 ~~ y4 + y6
    y3 ~~ y7
    y4 ~~ y8
    y6 ~~ y8
'
fitsem <- sem(model, data=dat2, meanstructure=TRUE)
fitsemaux <- auxiliary(fitsem, aux="z", data=dat2, meanstructure=TRUE)

#########################################################
```

```
## These following codes show the models that do not work with the current function

##### 1. covariate at the factor level
## HS.model.cov <- ' visual  =~ x1 + x2 + x3
##              textual =~ x4 + x5 + x6
##             speed   =~ x7 + x8 + x9
##   visual ~ sex
##   textual ~ sex
##   speed ~ sex'

## fitcov <- cfa(HS.model.cov, data=dat)
## fitcovaux <- auxiliary(fitcov, aux="z", data=dat)

### The auxiliary code does not work when specifying manually.
## HS.model.covxx <- ' visual  =~ x1 + x2 + x3
##              textual =~ x4 + x5 + x6
##              speed   =~ x7 + x8 + x9
##   visual ~ sex
##   textual ~ sex
##   speed ~ sex
##   z ~~ z
##   z ~~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9
##   z ~ sex'

## fitcovxx <- cfa(HS.model.covxx, data=dat)
## fitcovaux <- auxiliary(fitcov, aux="z", data=dat)

##### 2. Endogenous variable with single indicator
## HS.model.cov2 <- ' visual  =~ x1 + x2 + x3
##               textual =~ x4 + x5 + x6
##               x7 ~ visual + textual'
##
## fitcov2 <- sem (HS.model.cov2, data=dat, fixed.x=FALSE) #, group="sex", meanstructure=TRUE)
## fitcov2aux <- auxiliary(fitcov2, aux="z", data=dat)


### The auxiliary code does not work when specifying manually.
## HS.model.covyy <- ' visual  =~ x1 + x2 + x3
##               textual =~ x4 + x5 + x6
##               x7 ~ visual + textual
##    z ~~ x1 + x2 + x3 + x4 + x5 + x6 + x7'
## fitcovyy <- sem(HS.model.covyy, data=dat) #, group="sex", meanstructure=TRUE)
```

`clipboard_saveFile` *Copy or save the result of* `lavaan` *object into a clipboard or a file*

Description

Copy or save the result of `lavaan` object into a clipboard or a file. From the clipboard, users may paste the result into the Microsoft Excel or spreadsheet application to create a table of the output.

Usage

```
clipboard(object, what="summary", ...)
saveFile(object, file, what="summary", tableFormat=FALSE, ...)
```

Arguments

object	The lavaan object
what	The attributes of the lavaan object to be copied in the clipboard. "summary" is to copy the screen provided from the summary function. "mifit" is to copy the result from the miPowerFit function. Other attributes listed in the inspect method in the lavaan-class could also be used, such as "coef", "se", "fit", "samp", and so on.
file	A file name used for saving the result
tableFormat	If TRUE, save the result in the table format using tabs for seperation. Otherwise, save the result as the output screen printed in the R console.
...	Additional argument listed in the miPowerFit function.

Value

The resulting output will be saved into a clipboard or a file. If using the clipboard function, users may paste it in the other applications.

Author(s)

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

Examples

```
## Not run:
library(lavaan)
HW.model <- ' visual  =~ x1 + c1*x2 + x3
              textual =~ x4 + c1*x5 + x6
               speed   =~ x7 + x8 + x9 '

fit <- cfa(HW.model, data=HolzingerSwineford1939, group="school", meanstructure=TRUE)

# Copy the summary of the lavaan object
clipboard(fit)

# Copy the modification indices and the model fit from the miPowerFit function
clipboard(fit, "mifit")

# Copy the parameter estimates
clipboard(fit, "coef")

# Copy the standard errors
clipboard(fit, "se")

# Copy the sample statistics
clipboard(fit, "samp")

# Copy the fit measures
clipboard(fit, "fit")

# Save the summary of the lavaan object
saveFile(fit, "out.txt")

# Save the modification indices and the model fit from the miPowerFit function
saveFile(fit, "out.txt", "mifit")
```

```
# Save the parameter estimates
saveFile(fit, "out.txt", "coef")

# Save the standard errors
saveFile(fit, "out.txt", "se")

# Save the sample statistics
saveFile(fit, "out.txt", "samp")

# Save the fit measures
saveFile(fit, "out.txt", "fit")

## End(Not run)
```

`dat2way` *Simulated Dataset to Demonstrate Two-way Latent Interaction*

Description

A simulated data set with 2 independent factors and 1 dependent factor where each factor has three indicators

Usage

```
data(dat2way)
```

Format

A data frame with 500 observations of 9 variables.

x1 The first indicator of the first independent factor

x2 The second indicator of the first independent factor

x3 The third indicator of the first independent factor

x4 The first indicator of the second independent factor

x5 The second indicator of the second independent factor

x6 The third indicator of the second independent factor

x7 The first indicator of the dependent factor

x8 The second indicator of the dependent factor

x9 The third indicator of the dependent factor

Source

Data was generated by the mvrnorm function in the `MASS` package.

Examples

```
head(dat2way)
```

dat3way *Simulated Dataset to Demonstrate Three-way Latent Interaction*

Description

A simulated data set with 3 independent factors and 1 dependent factor where each factor has three indicators

Usage

```
data(dat3way)
```

Format

A data frame with 500 observations of 12 variables.

x1 The first indicator of the first independent factor

x2 The second indicator of the first independent factor

x3 The third indicator of the first independent factor

x4 The first indicator of the second independent factor

x5 The second indicator of the second independent factor

x6 The third indicator of the second independent factor

x7 The first indicator of the third independent factor

x8 The second indicator of the third independent factor

x9 The third indicator of the third independent factor

x10 The first indicator of the dependent factor

x11 The second indicator of the dependent factor

x12 The third indicator of the dependent factor

Source

Data was generated by the mvrnorm function in the MASS package.

Examples

```
head(dat3way)
```

exLong *Simulated Data set to Demonstrate Longitudinal Measurement Invariance*

Description

A simulated data set with 1 factors with 3 indicators in three timepoints

Usage

```
data(exLong)
```

Format

A data frame with 200 observations of 10 variables.

sex Sex of respondents

y1t1 Indicator 1 in Time 1

y2t1 Indicator 2 in Time 1

y3t1 Indicator 3 in Time 1

y1t2 Indicator 1 in Time 2

y2t2 Indicator 2 in Time 2

y3t2 Indicator 3 in Time 2

y1t3 Indicator 1 in Time 3

y2t3 Indicator 2 in Time 3

y3t3 Indicator 3 in Time 3

Source

Data was generated using the `simsem` package.

Examples

```
head(exLong)
```

findRMSEApower *Find the statistical power based on population RMSEA*

Description

Find the proportion of the samples from the sampling distribution of RMSEA in the alternative hypothesis rejected by the cutoff dervied from the sampling distribution of RMSEA in the null hypothesis. This function can be applied for both test of close fit and test of not-close fit (MacCallum, Browne, & Suguwara, 1996)

Usage

```
findRMSEApower(rmsea0, rmseaA, df, n, alpha=.05, group=1)
```

Arguments

rmsea0	Null RMSEA
rmseaA	Alternative RMSEA
df	Model degrees of freedom
n	Sample size of a dataset
alpha	Alpha level used in power calculations
group	The number of group that is used to calculate RMSEA.

Details

This function find the proportion of sampling distribution derived from the alternative RMSEA that is in the critical region derived from the sampling distribution of the null RMSEA. If rmseaA is greater than rmsea0, the test of close fit is used and the critical region is in the right hand side of the null sampling distribution. On the other hand, if rmseaA is less than rmsea0, the test of not-close fit is used and the critical region is in the left hand side of the null sampling distribution (MacCallum, Browne, & Suguwara, 1996).

Author(s)

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

References

MacCallum, R. C., Browne, M. W., & Sugawara, H. M. (1996). Power analysis and determination of sample size for covariance structure modeling. *Psychological Methods, 1,* 130-149.

See Also

- plotRMSEApower to plot the statistical power based on population RMSEA given the sample size
- plotRMSEAdist to visualize the RMSEA distributions
- findRMSEAsamplesize to find the minium sample size for a given statistical power based on population RMSEA

Examples

```
findRMSEApower(rmsea0=.05, rmseaA=.08, df=20, n=200)
```

---

findRMSEAsamplesize *Find the minimum sample size for a given statistical power based on population RMSEA*

Description

Find the minimum sample size for a specified statistical power based on population RMSEA. This function can be applied for both test of close fit and test of not-close fit (MacCallum, Browne, & Suguwara, 1996)

Usage

```
findRMSEAsamplesize(rmsea0, rmseaA, df, power=0.80, alpha=.05, group=1)
```

Arguments

rmsea0	Null RMSEA
rmseaA	Alternative RMSEA
df	Model degrees of freedom
power	Desired statistical power to reject misspecified model (test of close fit) or retain good model (test of not-close fit)
alpha	Alpha level used in power calculations
group	The number of group that is used to calculate RMSEA.

Details

This function find the minimum sample size for a specified power based on an iterative routine. The sample size keep increasing until the calculated power from `findRMSEApower` function is just over the specified power. If `group` is greater than 1, the resulting sample size is the sample size per group.

Author(s)

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

References

MacCallum, R. C., Browne, M. W., & Sugawara, H. M. (1996). Power analysis and determination of sample size for covariance structure modeling. *Psychological Methods, 1,* 130-149.

See Also

- `plotRMSEApower` to plot the statistical power based on population RMSEA given the sample size
- `plotRMSEAdist` to visualize the RMSEA distributions
- `findRMSEApower` to find the statistical power based on population RMSEA given a sample size

Examples

```
findRMSEAsamplesize(rmsea0=.05, rmseaA=.08, df=20, power=0.80)
```

indProd *Make products of indicators using no centering, mean centering, double-mean centering, or residual centering*

Description

The indProd function will make products of indicators using no centering, mean centering, double-mean centering, or residual centering. The orthogonalize function is the shortcut of the indProd function to make the residual-centered indicators products.

Usage

```
indProd(data, var1, var2, var3=NULL, match = TRUE, meanC = TRUE,
residualC = FALSE, doubleMC = TRUE, namesProd = NULL)
orthogonalize(data, var1, var2, var3=NULL, match=TRUE, namesProd=NULL)
```

Arguments

data	The desired data to be transformed.
var1	Names or indices of the variables loaded on the first factor
var2	Names or indices of the variables loaded on the second factor
var3	Names or indices of the variables loaded on the third factor (for three-way interaction)
match	Specify TRUE to use match-paired approach (Marsh, Wen, & Hau, 2004). If FALSE, the resulting products are all possible products.
meanC	Specify TRUE for mean centering the main effect indicator before making the products
residualC	Specify TRUE for residual centering the products by the main effect indicators (Little, Bovaird, & Widaman, 2006).
doubleMC	Specify TRUE for centering the resulting products (Lin et. al., 2010)
namesProd	The names of resulting products

Value

The original data attached with the products.

Author(s)

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

References

Marsh, H. W., Wen, Z. & Hau, K. T. (2004). Structural equation models of latent interactions: Evaluation of alternative estimation strategies and indicator construction. *Psychological Methods, 9,* 275-300.

Lin, G. C., Wen, Z., Marsh, H. W., & Lin, H. S. (2010). Structural equation models of latent interactions: Clarification of orthogonalizing and double-mean-centering strategies. *Structural Equation Modeling, 17*, 374-391.

Little, T. D., Bovaird, J. A., & Widaman, K. F. (2006). On the merits of orthogonalizing powered and product terms: Implications for modeling interactions among latent variables. *Structural Equation Modeling, 13*, 497-519.

See Also

- probe2WayMC For probing the two-way latent interaction when the results are obtained from mean-centering, or double-mean centering.
- probe3WayMC For probing the three-way latent interaction when the results are obtained from mean-centering, or double-mean centering.
- probe2WayRC For probing the two-way latent interaction when the results are obtained from residual-centering approach.
- probe3WayRC For probing the two-way latent interaction when the results are obtained from residual-centering approach.
- plotProbe Plot the simple intercepts and slopes of the latent interaction.

Examples

```
# Mean centering / two-way interaction / match-paired
dat <- indProd(attitude[,-1], var1=1:3, var2=4:6)

# Residual centering / two-way interaction / match-paired
dat2 <- indProd(attitude[,-1], var1=1:3, var2=4:6, match=FALSE, meanC=FALSE, residualC=TRUE, doubleMC=FALSE

# Double-mean centering / two-way interaction / match-paired
dat3 <- indProd(attitude[,-1], var1=1:3, var2=4:6, match=FALSE, meanC=TRUE, residualC=FALSE, doubleMC=TRUE)

# Mean centering / three-way interaction / match-paired
dat4 <- indProd(attitude[,-1], var1=1:2, var2=3:4, var3=5:6)

# Residual centering / three-way interaction / match-paired
dat5 <- indProd(attitude[,-1], var1=1:2, var2=3:4, var3=5:6, match=FALSE, meanC=FALSE, residualC=TRUE, doub

# Double-mean centering / three-way interaction / match-paired
dat6 <- indProd(attitude[,-1], var1=1:2, var2=3:4, var3=5:6, match=FALSE, meanC=TRUE, residualC=TRUE, doubl
```

---

kurtosis *Finding excessive kurtosis*

Description

Finding excessive kurtosis (g2) of an object

Usage

```
kurtosis(object, population=FALSE)
```

Arguments

object	A vector used to find a excessive kurtosis
population	TRUE to compute the parameter formula. FALSE to compute the sample statistic formula.

Details

The excessive kurtosis computed is g2. The parameter excessive kurtosis γ_2 formula is

$$\gamma_2 = \frac{\mu_4}{\mu_2^2} - 3,$$

where μ_i denotes the i order central moment.

The excessive kurtosis formula for sample statistic g_2 is

$$g_2 = \frac{k_4}{k_2^2},$$

where k_i are the i order k-statistic.

The standard error of the excessive kurtosis is

$$Var(\hat{g}_2) = \frac{24}{N}$$

where N is the sample size.

Value

A value of an excessive kurtosis with a test statistic if the population is specified as `FALSE`

Author(s)

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

References

Weisstein, Eric W. (n.d.). *Kurtosis.* Retrived from MathWorld–A Wolfram Web Resource http://mathworld.wolfram.com/Kurtosis.html

See Also

- `skew` Find the univariate skewness of a variable
- `mardiaSkew` Find the Mardia's multivariate skewness of a set of variables
- `mardiaKurtosis` Find the Mardia's multivariate kurtosis of a set of variables

Examples

```
kurtosis(1:5)
```

`lavaanStar-class` *Class For Representing A (Fitted) Latent Variable Model with Additional Elements*

Description

This is the `lavaan` class that contains additional information about the fit values from the null model. Some functions are adjusted according to the change.

Objects from the Class

Objects can be created via the `auxiliary` function or `runMI`.

Slots

`call`: The function call as returned by `match.called()`.

`timing`: The elapsed time (user+system) for various parts of the program as a list, including the total time.

`Options`: Named list of options that were provided by the user, or filled-in automatically.

`ParTable`: Named list describing the model parameters. Can be coerced to a data.frame. In the documentation, this is called the 'parameter table'.

`Data`: Object of internal class `"Data"`: information about the data.

`SampleStats`: Object of internal class `"SampleStats"`: sample statistics

`Model`: Object of internal class `"Model"`: the internal (matrix) representation of the model

`Fit`: Object of internal class `"Fit"`: the results of fitting the model

`nullfit`: The fit-indices information from the null model

`imputed`: The list of information from running multiple imputation. The first element is the convergence rate of the target and null models. The second element is the fraction missing information. The first estimate of FMI (FMI.1) is asymptotic FMI and the second estimate of FMI (FMI.2) is corrected for small numbers of imputation. The third element is the fit values of the target model by the specified chi-squared methods. The fourth element is the fit values of the null model by the specified chi-square methods.

References

see `linkS4class{lavaan}`

See Also

`auxiliary`; `runMI`

Examples

```
HS.model <- ' visual  =~ x1 + x2 + x3
              textual =~ x4 + x5 + x6
              speed   =~ x7 + x8 + x9 '

dat <- data.frame(HolzingerSwineford1939, z=rnorm(nrow(HolzingerSwineford1939), 0, 1))

fit <- cfa(HS.model, data=dat) #, group="sex", meanstructure=TRUE)
fitaux <- auxiliary(fit, aux="z", data=dat)
```

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)
```

longInvariance *Measurement Invariance Tests Within Person*

Description

Testing measurement invariance across timepoints (longitudinal) or any context involving the use of the same scale in one case (e.g., a dyad case with husband and wife answering the same scale). The measurement invariance uses a typical sequence of model comparison tests. This function currently works with only one scale.

Usage

```
longInvariance(model, varList, auto = "all", constrainAuto = FALSE,
fixed.x = TRUE, std.lv = FALSE, group=NULL, group.equal="",
group.partial="", warn=TRUE, debug=FALSE, strict = FALSE, quiet = FALSE,
...)
```

Arguments

model	lavaan syntax or parameter table
varList	A list containing indicator names of factors used in the invariance testing, such as the list that the first element is the vector of indicator names in the first timepoint and the second element is the vector of indicator names in the second timepoint. The order of indicator names should be the same (but measured in different times or different units).
auto	The order of autocorrelation on the measurement errors on the similar items across factor (e.g., Item 1 in Time 1 and Time 2). If 0 is specified, the autocorrelation will be not imposed. If 1 is specified, the autocorrelation will imposed for the adjacent factor listed in varList. The maximum number can be specified is the number of factors specified minus 1. If "all" is specified, the maximum number of order will be used.
constrainAuto	If TRUE, the function will equate the auto-*covariance* to be equal within the same item across factors. For example, the covariance of item 1 in time 1 and time 2 is equal to the covariance of item 1 in time 2 and time 3.
fixed.x	See lavaan.
std.lv	See lavaan.
group	See lavaan.
group.equal	See lavaan.
group.partial	See lavaan.
warn	See lavaan.
debug	See lavaan.
strict	If TRUE, the sequence requires 'strict' invariance. See details for more information.
quiet	If TRUE, a summary is printed out containing an overview of the different models that are fitted, together with some model comparison tests.
...	Additional arguments in the lavaan function.

Details

If strict = FALSE, the following four models are tested in order:

1. Model 1: configural invariance. The same factor structure is imposed on all units.
2. Model 2: weak invariance. The factor loadings are constrained to be equal across units.
3. Model 3: strong invariance. The factor loadings and intercepts are constrained to be equal across units.
4. Model 4: The factor loadings, intercepts and means are constrained to be equal across units.

Each time a more restricted model is fitted, a chi-square difference test is reported, comparing the current model with the previous one, and comparing the current model to the baseline model (Model 1). In addition, the difference in cfi is also reported (delta.cfi).

If strict = TRUE, the following five models are tested in order:

1. Model 1: configural invariance. The same factor structure is imposed on all units.
2. Model 2: weak invariance. The factor loadings are constrained to be equal across units.
3. Model 3: strong invariance. The factor loadings and intercepts are constrained to be equal across units.

4. Model 4: strict invariance. The factor loadings, intercepts and residual variances are constrained to be equal across units.
5. Model 5: The factor loadings, intercepts, residual variances and means are constrained to be equal across units.

Note that if the chi-square test statistic is scaled (eg. a Satorra-Bentler or Yuan-Bentler test statistic), a special version of the chi-square difference test is used as described in http://www.statmodel.com/chidiff.shtml

Value

Invisibly, all model fits in the sequence are returned as a list.

Author(s)

Sunthud Pornprasertmanit (University of Kansas; <psunthud@ku.edu>); Yves Rosseel (Ghent University; <Yves.Rosseel@UGent.be>)

References

Vandenberg, R. J., and Lance, C. E. (2000). A review and synthesis of the measurement invariance literature: Suggestions, practices, and recommendations for organizational research. *Organizational Research Methods, 3,* 4-70.

See Also

measurementinvariance For the measurement invariance test between groups

Examples

```
model <- ' f1t1 =~ y1t1 + y2t1 + y3t1
             f1t2 =~ y1t2 + y2t2 + y3t2
  f1t3 =~ y1t3 + y2t3 + y3t3'

# Create list of variables
var1 <- c("y1t1", "y2t1", "y3t1")
var2 <- c("y1t2", "y2t2", "y3t2")
var3 <- c("y1t3", "y2t3", "y3t3")
constrainedVar <- list(var1, var2, var3)

# Invariance of the same factor across timepoints
longInvariance(model, auto=1, constrainAuto=TRUE, varList=constrainedVar, data=exLong)

# Invariance of the same factor across timepoints and groups
longInvariance(model, auto=1, constrainAuto=TRUE, varList=constrainedVar, data=exLong, group="sex", group.e
```

`mardiaKurtosis` *Finding Mardia's multivariate kurtosis*

Description

Finding Mardia's multivariate kurtosis of multiple variables

Usage

```
mardiaKurtosis(dat)
```

Arguments

`dat`	The target matrix or data frame with multiple variables

Details

The Mardia's multivariate kurtosis formula (Mardia, 1970) is

$$b_{2,d} = \frac{1}{n}\sum_{i=1}^{n}\left[\left(\boldsymbol{X}_i - \bar{\boldsymbol{X}}\right)' \boldsymbol{S}^{-1}\left(\boldsymbol{X}_i - \bar{\boldsymbol{X}}\right)\right]^2,$$

where d is the number of variables, X is the target dataset with multiple variables, n is the sample size, $\boldsymbol{S}$ is the sample covariance matrix of the target dataset, and $\bar{\boldsymbol{X}}$ is the mean vectors of the target dataset binded in n rows. When the population multivariate kurtosis is normal, the $b_{2,d}$ is asymptotically distributed as normal distribution with the mean of $d(d+2)$ and variance of $8d(d+2)/n$.

Value

A value of a Mardia's multivariate kurtosis with a test statistic

Author(s)

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

References

Mardia, K. V. (1970). Measures of multivariate skewness and kurtosis with applications. *Biometrika, 57*, 519-530.

See Also

- `skew` Find the univariate skewness of a variable
- `kurtosis` Find the univariate excessive kurtosis of a variable
- `mardiaSkew` Find the Mardia's multivariate skewness of a set of variables

Examples

```
library(lavaan)
mardiaKurtosis(HolzingerSwineford1939[,paste("x", 1:9, sep="")])
```

mardiaSkew *Finding Mardia's multivariate skewness*

Description

Finding Mardia's multivariate skewness of multiple variables

Usage

```
mardiaSkew(dat)
```

Arguments

dat	The target matrix or data frame with multiple variables

Details

The Mardia's multivariate skewness formula (Mardia, 1970) is

$$b_{1,d} = \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} \left[\left(\boldsymbol{X}_i - \bar{\boldsymbol{X}} \right)' \boldsymbol{S}^{-1} \left(\boldsymbol{X}_j - \bar{\boldsymbol{X}} \right) \right]^3 ,$$

where d is the number of variables, X is the target dataset with multiple variables, n is the sample size, $\boldsymbol{S}$ is the sample covariance matrix of the target dataset, and $\bar{\boldsymbol{X}}$ is the mean vectors of the target dataset binded in n rows. When the population multivariate skewness is normal, the $\frac{n}{6} b_{1,d}$ is asymptotically distributed as chi-square distribution with $d(d+1)(d+2)/6$ degrees of freedom.

Value

A value of a Mardia's multivariate skewness with a test statistic

Author(s)

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

References

Mardia, K. V. (1970). Measures of multivariate skewness and kurtosis with applications. *Biometrika, 57*, 519-530.

See Also

- skew Find the univariate skewness of a variable
- kurtosis Find the univariate excessive kurtosis of a variable
- mardiaKurtosis Find the Mardia's multivariate kurtosis of a set of variables

Examples

```
library(lavaan)
mardiaSkew(HolzingerSwineford1939[,paste("x", 1:9, sep="")])
```

`measurementInvariance` *Measurement Invariance Tests*

Description

Testing measurement invariance across groups using a typical sequence of model comparison tests.

Usage

```
measurementInvariance(..., strict = FALSE, quiet = FALSE)
```

Arguments

`...`	The same arguments as for any lavaan model. See `cfa` for more information.
`strict`	If `TRUE`, the sequence requires 'strict' invariance. See details for more information.
`quiet`	If `TRUE`, a summary is printed out containing an overview of the different models that are fitted, together with some model comparison tests.

Details

If `strict = FALSE`, the following four models are tested in order:

1. Model 1: configural invariance. The same factor structure is imposed on all groups.
2. Model 2: weak invariance. The factor loadings are constrained to be equal across groups.
3. Model 3: strong invariance. The factor loadings and intercepts are constrained to be equal across groups.
4. Model 4: The factor loadings, intercepts and means are constrained to be equal across groups.

Each time a more restricted model is fitted, a chi-square difference test is reported, comparing the current model with the previous one, and comparing the current model to the baseline model (Model 1). In addition, the difference in cfi is also reported (delta.cfi).

If `strict = TRUE`, the following five models are tested in order:

1. Model 1: configural invariance. The same factor structure is imposed on all groups.
2. Model 2: weak invariance. The factor loadings are constrained to be equal across groups.
3. Model 3: strong invariance. The factor loadings and intercepts are constrained to be equal across groups.
4. Model 4: strict invariance. The factor loadings, intercepts and residual variances are constrained to be equal across groups.
5. Model 5: The factor loadings, intercepts, residual variances and means are constrained to be equal across groups.

Note that if the chi-square test statistic is scaled (eg. a Satorra-Bentler or Yuan-Bentler test statistic), a special version of the chi-square difference test is used as described in http://www.statmodel.com/chidiff.shtml

Value

Invisibly, all model fits in the sequence are returned as a list.

Author(s)

Yves Rosseel <Yves.Rosseel@UGent.be>

References

Vandenberg, R. J., and Lance, C. E. (2000). A review and synthesis of the measurement invariance literature: Suggestions, practices, and recommendations for organizational research. *Organizational Research Methods, 3,* 4-70.

See Also

`longInvariance` For the measurement invariance test within person

Examples

```
HW.model <- ' visual =~ x1 + x2 + x3
              textual =~ x4 + x5 + x6
              speed =~ x7 + x8 + x9 '

measurementInvariance(HW.model, data=HolzingerSwineford1939, group="school")
```

---

miPowerFit *Modification indices and their power approach for model fit evaluation*

Description

The model fit evaluation approach using modification indices and their power proposed by Saris, Satorra, and van der Veld (2009, pp. 570-573).

Usage

```
miPowerFit(lavaanObj, stdLoad=0.4, cor=0.1, stdBeta=0.1, intcept=0.2, stdDelta=NULL, delta=NULL)
```

Arguments

`lavaanObj`	The lavaan model object used to evaluate model fit
`stdLoad`	The amount of standardized factor loading that one would like to be detected (rejected). The default value is 0.4, which is suggested by Saris and colleagues (2009, p. 571).
`cor`	The amount of factor or error correlations that one would like to be detected (rejected). The default value is 0.1, which is suggested by Saris and colleagues (2009, p. 571).
`stdBeta`	The amount of standardized regression coefficients that one would like to be detected (rejected). The default value is 0.1, which is suggested by Saris and colleagues (2009, p. 571).
`intcept`	The amount of standardized intercept (similar to Cohen's *d* that one would like to be detected (rejected). The default value is 0.2, which is equivalent to a low effect size proposed by Cohen (1988, 1992).

`stdDelta` The vector of the standardized parameters that one would like to be detected (rejected). If this argument is specified, the value here will overwrite the other arguments above. The order of the vector must be the same as the row order from modification indices from the `lavaan` object. If a single value is specified, the value will be applied to all parameters.

`delta` The vector of the unstandardized parameters that one would like to be detected (rejected). If this argument is specified, the value here will overwrite the other arguments above. The order of the vector must be the same as the row order from modification indices from the `lavaan` object. If a single value is specified, the value will be applied to all parameters.

Details

In the lavaan object, one can inspect the modification indices and expected parameter changes. Those values can be used to evaluate model fit by the method proposed by Saris and colleagues (2009). First, one should evaluate whether the modification index of each parameter is significant. Second, one should evaluate whether the power to detect a target expected parameter change is high enough. If the modification index is not significant and the power is high, there is no misspecification. If the modification index is significant and the power is low, the fixed parameter is misspecified. If the modification index is significant and the power is high, the expected parameter change is investigated. If the expected parameter change is large (greater than the the target expected parameter change), the parameter is misspecified. If the expected parameter change is low (lower than the target expected parameter change), the parameter is not misspecificied. If the modification index is not significant and the power is low, the decision is inconclusive.

Value

A data frame with these variables:

1. lhs The left-hand side variable (with respect to the lavaan operator)
2. op The lavaan syntax operator: "~~" represents covariance, "=~" represents factor loading, "~" represents regression, and "~1" represents intercept.
3. rhs The right-hand side variable (with respect to the lavaan operator)
4. group The group of the parameter
5. mi The modification index of the fixed parameter
6. epc The expected parameter change if the parameter is freely estimated
7. target.epc The target expected parameter change that represents the minimum size of misspecification that one would like to be detected by the test with a high power
8. std.epc The standardized expected parameter change if the parameter is freely estimated
9. std.target.epc The standardized target expected parameter change
10. significant.mi Represents whether the modification index value is significant
11. high.power Represents whether the power is enough to detect the target expected parameter change
12. decision The decision whether the parameter is misspecified or not: `"M"` represents the parameter is misspecified, `"NM"` represents the parameter is not misspecified, `"EPC:M"` represents the parameter is misspecified decided by checking the expected parameter change value, `"EPC:NM"` represents the parameter is not misspecified decided by checking the expected parameter change value, and `"I"` represents the decision is inconclusive.

Author(s)

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

References

Cohen, J. (1988). *Statistical power analysis for the behavioral sciences* (2nd ed.). Hillsdale, NJ: Erlbaum.

Cohen, J. (1992). A power primer. *Psychological Bulletin, 112*, 155-159.

Saris, W. E., Satorra, A., & van der Veld, W. M. (2009). Testing structural equation models or detection of misspecifications? *Structural Equation Modeling, 16*, 561-582.

See Also

`moreFitIndices` For the additional fit indices information

Examples

```
library(lavaan)

HS.model <- ' visual  =~ x1 + x2 + x3
              textual =~ x4 + x5 + x6
              speed   =~ x7 + x8 + x9 '

fit <- cfa(HS.model, data=HolzingerSwineford1939, group="sex", meanstructure=TRUE)
miPowerFit(fit)

model <- '
  # latent variable definitions
     ind60 =~ x1 + x2 + x3
     dem60 =~ y1 + a*y2 + b*y3 + c*y4
     dem65 =~ y5 + a*y6 + b*y7 + c*y8

  # regressions
    dem60 ~ ind60
    dem65 ~ ind60 + dem60

  # residual correlations
    y1 ~~ y5
    y2 ~~ y4 + y6
    y3 ~~ y7
    y4 ~~ y8
    y6 ~~ y8
'
fit2 <- sem(model, data=PoliticalDemocracy, meanstructure=TRUE)
miPowerFit(fit2, stdLoad=0.3, cor=0.2, stdBeta=0.2, intcept=0.5)
```

`monteCarloMed` *Monte Carlo Confidence Intervals to Test Complex Indirect Effects*

Description

This function takes an expression for an indirect effect, the parameters and standard errors associated with the expression and returns a confidence interval based on a Monte Carlo test of mediation (MacKinnon, Lockwood, & Williams, 2004).

Usage

```
monteCarloMed(expression, ..., ACM=NULL, rep=20000, CI=95, plot=FALSE, outputValues=FALSE)
```

Arguments

`expression`	A character scalar representing the computation of an indirect effect. Different parameters in the expression should have different alphanumeric values. Expressions can use either addition (+) or multiplication (*) operators.
`...`	Parameter estimates for all parameters named in `expression`. The order of parameters should follow from `expression` (the first parameter named in `expression` should be the first parameter listed in ...). Alternatively ...can be a vector of parameter estimates.
`ACM`	A matrix representing the asymptotic covariance matrix of the parameters described in `expression`. This matrix should be a symetric matrix with dimensions equal to the number of parameters names in `expression`. Information on finding the ACOV is popular SEM software is described below.)
`rep`	The number of replications to compute. Many thousand are reccomended.
`CI`	Width of the confidence interval computed.
`plot`	Should the function output a plot of simulated values of the indirect effect?
`outputValues`	Should the function output all simulated values of the indirect effect?

Details

This function implements the Monte Carlo test of mediation first described in MacKinnon, Lockwood, & Williams (2004) and extends it to complex cases where the indirect effect is more than a function of two parameters. The function takes an expression for the indirect effect, randomly simulated values of the indirect effect based on the values of the parameters (and the associated standard errors) comprising the indirect effect, and outputs a confidence interval of the indirect effect based on the simulated values. For further information on the Monte Carlo test of mediation see MacKinnon, Lockwood, & Williams (2004), Preacher & Selig (in press), and Selig & Preacher (2008). For a Monte Carlo test of mediation with a random effects model see Selig & Preacher (2010).

The asymptotic covariance matrix can be easily found in many popular SEM software applications.

- LISRELIncluding the EC option on the OU line will print the ACM to a seperate file. The file contains the lower triangular elements of the ACM in free format and scientific notation
- MplusInclude the command TECH3; in the OUTPUT section. The ACM will be printed in the output.
- lavaan Use the command `vcov` on the fitted lavaan object to print the ACM to the screen

Value

A list with two elements. The first element is the point estimate for the indirect effect. The second element is a matrix with values for the upper and lower limits of the confidence interval generated from the Monte Carlo test of mediation. If `outputValues=TRUE`, output will be a list with a list with the point estimate and values for the upper and lower limits of the confidence interval as the first element and a vector of simulated values of the indirect effect as the second element.

Author(s)

Corbin Quick (University of Kansas; <corbinq@ku.edu>) Alexander M. Schoemann (University of Kansas; <schoemann@ku.edu>) James P. Selig (University of New Mexico; <selig@unm.edu>)

References

Preacher, K. J., & Selig, J. P. (2010, July). Monte Carlo method for assessing multilevel mediation: An interactive tool for creating confidence intervals for indirect effects in 1-1-1 multilevel models [Computer software]. Available from http://quantpsy.org/.

Preacher, K. J., & Selig, J. P. (in press). Advantages of Monte Carlo confidence intervals for indirect effects. *Communication Methods and Measures*.

Selig, J. P., & Preacher, K. J. (2008, June). Monte Carlo method for assessing mediation: An interactive tool for creating confidence intervals for indirect effects [Computer software]. Available from http://quantpsy.org/.

Examples

```
#Simple two path mediation
#Write expression of indirect effect
med <- 'a*b'
#Paramter values from analyses
aparam <- 1
bparam<-2
#Asymptotic covariance matrix from analyses
AC <- matrix(c(.01,.00002,
               .00002,.02), nrow=2, byrow=TRUE)
#Compute CI, include a plot
monteCarloMed(med, coef1=aparam, coef2=bparam, outputValues=FALSE, plot=TRUE, ACM=AC)

#Use a matrix of parameter estimates as input
aparam<-c(1,2)
monteCarloMed(med, coef1=aparam, outputValues=FALSE, plot=TRUE, ACM=AC)



#complex mediation with two paths for the indirect effect
#Write expression of indirect effect
med <- 'a1*b1 + a1*b2'
#Paramter values and standard errors from analyses
aparam <- 1
b1param<-2
b2param<-1
#Asymptotic covariance matrix from analyses
AC <- matrix(c(1,.00002, .00003,
                    .00002,1, .00002,
.00003, .00002, 1), nrow=3, byrow=TRUE)
#Compute CI do not include a plot
monteCarloMed(med, coef1=aparam, coef2=b1param, coef3=b2param, ACM=AC)
```

`moreFitIndices` *Calculate more fit indices*

Description

Calculate more fit indices that are not already provided in lavaan.

Usage

```
moreFitIndices(object, nPrior = 1)
```

Arguments

`object`	The lavaan model object provided after running the `cfa` or the `sem` functions.
`nPrior`	The sample size on which prior is based. This argument is used to compute BIC*.

Details

Normed Fit Index (nfi; West, Taylor, & Wu, 2012) is one of the relative fit indices which can be computed by

$$nfi = \frac{\chi_0^2 - \chi_k^2}{\chi_0^2},$$

where χ_k^2 is the chi-square test statistic value of the target model, χ_0^2 is the chi-square test statistic value of the null model.

Incremental Fit Index (ifi; West, Taylor, & Wu, 2012) is one of the relative fit indices which can be computed by

$$ifi = \frac{\chi_0^2 - \chi_k^2}{\chi_0^2 - df_k},$$

where df_k is the degree of freedom when fitting the target model

Gamma Hat (gfi*; West, Taylor, & Wu, 2012) is one of the relative fit indices which can be computed by

$$gfi* = \frac{p}{p + 2 \times \frac{\chi_k^2 - df_k}{N-1}},$$

where N is the sample size, p is the number of variables in the model. This formula assumes equal number of indicators across groups.

Adjusted Gamma Hat (agfi*; West, Taylor, & Wu, 2012) is one of the relative fit indices which can be computed by

$$agfi* = \left(1 - \frac{K \times p \times (p+1)}{2 \times df_k}\right) \times (1 - gfi*),$$

where K is the number of group (please refer to Dudgeon, 2004 for the multiple-group adjustment for agfi*).

Corrected Akaike Information Criterion (AICc; Burnham & Anderson, 2003) is the corrected version of aic for small sample size:

$$aicc = f + \frac{2k(k+1)}{N-k-1},$$

where f is the minimized discrepancy function, which is the product of the log likelihood and -2, and k is the number of parameters in the target model.

Expected Value of Cross-Validation Index (ECVI; West, Taylor, & Wu, 2012) is the average discrepancy in the fitted covariance matrices between two samples of equal sample size across all possible combinations of two samples from the same population:

$$ecvi = f + \frac{2 \times k}{N},$$

Stochastic information criterion (sic; Preacher, 2006) is similar to aic or bic. This index will account for model complexity in the model's function form, in addition to the number of free parameters. This index will be provided only the chi-square value is not scaled. sic can be computed by

$$sic = \frac{1}{2}\left(f - \log\det I(\hat{\theta})\right),$$

where $I(\hat{\theta})$ is the information matrix of the parameters.

Corrected Bayesian Information Criterion (BIC*; Kuha, 2004) is similar to bic but explicitly specifying the sample size on which the prior is based (N_{prior}).

$$bicc = f + k\log\left(1 + N/N_{prior}\right),$$

Hannan-Quinn Information Criterion (hqc; Hannan & Quinn, 1979) is used for model selection similar to aic or bic.

$$hqc = f + 2k\log\left(\log N\right),$$

Note that if Satorra-Bentler or Yuan-Bentler's method is used, the fit indices using the scaled chi-square values are also provided.

Value

1. nfi Normed Fit Index
2. ifi Incremental Fit Index
3. gfi* Gamma Hat
4. agfi* Adjusted Gamma Hat
5. aicc Corrected Akaike Information Criterion
6. ecvi Expected Value of Cross-Validation Index
7. sic Stochastic Information Criterion
8. bic* Bayesian Information Criterion with specifying the prior sample size
9. hqc Hannan-Quinn Information Criterion
10. nfi.scaled Normed Fit Index using Scaled Chi-square
11. ifi.scaled Incremental Fit Index using Scaled Chi-square
12. gfi*.scaled Gamma Hat using Scaled Chi-square
13. agfi*.scaled Adjusted Gamma Hat using Scaled Chi-square

Author(s)

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

References

Burnham, K., & Anderson, D. (2003). *Model selection and multimodel inference: A practical-theoretic approach.* New York, NY: Springer-Verlag.

Dudgeon, P. (2004). A note on extending Steiger's (1998) multiple sample RMSEA adjustment to other noncentrality parameter-based statistic. *Structural Equation Modeling, 11*, 305-319.

Kuha, J. (2004). AIC and BIC: Comparisons of assumptions and performance. *Sociological Methods Research, 33,* 188-229.

Preacher, K. J. (2006). Quantifying parsimony in structural equation modeling. *Multivariate Behavioral Research, 43,* 227-259.

West, S. G., Taylor, A. B., & Wu, W. (2012). Model fit and model selection in structural equation modeling. In R. H. Hoyle (Ed.), *Handbook of Structural Equation Modeling.* New York: Guilford.

See Also

`miPowerFit` For the modification indices and their power approach for model fit evaluation

Examples

```
HS.model <- ' visual  =~ x1 + x2 + x3
              textual =~ x4 + x5 + x6
              speed   =~ x7 + x8 + x9 '

fit <- cfa(HS.model, data=HolzingerSwineford1939)
moreFitIndices(fit)

fit2 <- cfa(HS.model, data=HolzingerSwineford1939, estimator="mlr")
moreFitIndices(fit2)
```

---

parcelAllocation *Random Allocation of Items to Parcels in a Structural Equation Model*

Description

This function generates a given number of randomly generated item-to-parcel allocations, fits a model to each allocation, and provides averaged results over all allocations.

Usage

```
parcelAllocation(nPerPar, facPlc, nAlloc=100, syntax, dataset, names='default', leaveout=0, ...)
```

Arguments

nPerPar	A list in which each element is a vector corresponding to each factor indicating sizes of parcels. If variables are left out of parceling, they should not be accounted for here (there should NOT be parcels of size "1").
facPlc	A list of vectors, each corresponding to a factor, specifying the variables in that factor (whether included in parceling or not). Either variable names or column numbers. Variables not listed will not be modeled or included in output datasets.
nAlloc	The number of random allocations of items to parcels to generate.
syntax	lavaan syntax. If substituted with a file name, parcelAllocation will print output data sets to a specified folder rather than analyzing using lavaan (note for Windows users: file path must be specified using forward slashes).
dataset	Data set. Can be file path or R object (matrix or dataframe). If the data has missing values multiple imputation before parceling is recommended.
names	(Optional) A character vector containing the names of parceled variables.
leaveout	A vector of variables to be left out of randomized parceling. Either variable names or column numbers are allowed.
...	Additional arguments to be passed to lavaan

Details

This function implements the random item to parcel allocation procedure described in Sterba (2011) and Sterba and McCallum (2010). The function takes a single data set with item level data, randomly assigns items to parcels, fits a structural equation model to the parceled data (using lavaan), and repeats this process for a user specified number of random allocations. Results from all fitted models are summarized and output. For further details on the benefits of the random allocation of itesm to parcels see Sterba (2011) and Sterba and McCallum (2010).

Value

Estimates	A data frame containing results related to parameter estimates with columns corresponding to parameter names, average parameter estimates across allocations, the standard deviation of parameter estimates across allocations, the minimum parameter estimate across allocations, the maximum parameter estimate across allocations, the range of parameter estimates across allocations, and the proportions of allocations in which the parameter estimate is significant.
SE	A data frame containing results related to standard errors with columns corresponding to parameter names, average standard errors across allocations, the standard deviation of standard errors across allocations, the minimum standard error across allocations, the maximum standard error across allocations, and the range of standard errors across allocations.
Fit	A data frame containing results related to model fit with columns corresponding to fit index names, the average of each index across allocations, the standard deviation of each fit index across allocations, the minimum of each fit index across allocations, the maximum of each fit index across allocations, and the range of each fit index across allocations.

Author(s)

Corbin Quick (University of Kansas; <corbinq@ku.edu>) Alexander M. Schoemann (University of Kansas; <schoemann@ku.edu>)

References

Sterba, S.K. (2011). Implications of parcel-allocation variability for comparing fit of item-solutions and parcel-solutions. *Structural Equation Modeling, 18,* 554-577.

Sterba, S.K. & MacCallum, R.C. (2010). Variability in parameter estimates and model fit across random allocations of items to parcels. *Multivariate Behavioral Research, 45,* 322-358.

Examples

```
#Fit 3 factor CFA to simulated data.
#Each factor has 9 indicators that are randomly parceled into 3 parcels
#Lavaan syntax for the model to be fit to parceled data
syntax <- 'La =~ V1 + V2 + V3
           Lb =~ V4 + V5 + V6
'
#Parcel and fit data 20 times. The actual parcel number should be higher than 20 times.
name1 <- colnames(simParcel)[1:9]
name2 <- colnames(simParcel)[10:18]
parcelAllocation(list(c(3,3,3),c(3,3,3)), list(name1, name2), nAlloc=20, syntax=syntax, dataset=simParcel)
```

---

plotProbe *Plot the graphs for probing latent interaction*

Description

This function will plot the line graphs representing the simple effect of the independent variable given the values of the moderator.

Usage

```
plotProbe(object, xlim, xlab="Indepedent Variable", ylab="Dependent Variable", ...)
```

Arguments

`object`	The result of probing latent interaction obtained from `probe2WayMC`, `probe2WayRC`, `probe3WayMC`, or `probe3WayRC` function.
`xlim`	The vector of two numbers: the minimum and maximum values of the independent variable
`xlab`	The label of the x-axis
`ylab`	The label of the y-axis
`...`	Any addition argument for the `plot` function

Value

None. This function will plot the simple main effect only.

Author(s)

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

See Also

- indProd For creating the indicator products with no centering, mean centering, double-mean centering, or residual centering.
- probe2WayMC For probing the two-way latent interaction when the results are obtained from mean-centering, or double-mean centering.
- probe3WayMC For probing the three-way latent interaction when the results are obtained from mean-centering, or double-mean centering.
- probe2WayRC For probing the two-way latent interaction when the results are obtained from residual-centering approach.
- probe3WayRC For probing the two-way latent interaction when the results are obtained from residual-centering approach.

Examples

```
library(lavaan)

dat2wayMC <- indProd(dat2way, 1:3, 4:6)

model1 <- "
f1 =~ x1 + x2 + x3
f2 =~ x4 + x5 + x6
f12 =~ x1.x4 + x2.x5 + x3.x6
f3 =~ x7 + x8 + x9
f3 ~ f1 + f2 + f12
f12 ~~0*f1
f12 ~~ 0*f2
x1 ~ 0*1
x4 ~ 0*1
x1.x4 ~ 0*1
x7 ~ 0*1
f1 ~ NA*1
f2 ~ NA*1
f12 ~ NA*1
f3 ~ NA*1
"

fitMC2way <- sem(model1, data=dat2wayMC, meanstructure=TRUE, std.lv=FALSE)
result2wayMC <- probe2WayMC(fitMC2way, c("f1", "f2", "f12"), "f3", "f2", c(-1, 0, 1))
plotProbe(result2wayMC, xlim=c(-2, 2))


dat3wayMC <- indProd(dat3way, 1:3, 4:6, 7:9)

model3 <- "
f1 =~ x1 + x2 + x3
f2 =~ x4 + x5 + x6
f3 =~ x7 + x8 + x9
f12 =~ x1.x4 + x2.x5 + x3.x6
f13 =~ x1.x7 + x2.x8 + x3.x9
f23 =~ x4.x7 + x5.x8 + x6.x9
f123 =~ x1.x4.x7 + x2.x5.x8 + x3.x6.x9
f4 =~ x10 + x11 + x12
f4 ~ f1 + f2 + f3 + f12 + f13 + f23 + f123
f1 ~~ 0*f12
```

```
f1 ~~ 0*f13
f1 ~~ 0*f123
f2 ~~ 0*f12
f2 ~~ 0*f23
f2 ~~ 0*f123
f3 ~~ 0*f13
f3 ~~ 0*f23
f3 ~~ 0*f123
f12 ~~ 0*f123
f13 ~~ 0*f123
f23 ~~ 0*f123
x1 ~ 0*1
x4 ~ 0*1
x7 ~ 0*1
x10 ~ 0*1
x1.x4 ~ 0*1
x1.x7 ~ 0*1
x4.x7 ~ 0*1
x1.x4.x7 ~ 0*1
f1 ~ NA*1
f2 ~ NA*1
f3 ~ NA*1
f12 ~ NA*1
f13 ~ NA*1
f23 ~ NA*1
f123 ~ NA*1
f4 ~ NA*1
"

fitMC3way <- sem(model3, data=dat3wayMC, meanstructure=TRUE, std.lv=FALSE)
result3wayMC <- probe3WayMC(fitMC3way, c("f1", "f2", "f3", "f12", "f13", "f23", "f123"), "f4", c("f1", "f2"
plotProbe(result3wayMC, xlim=c(-2, 2))
```

`plotRMSEAdist` *Plot the sampling distributions of RMSEA*

Description

Plots the sampling distributions of RMSEA based on the noncentral chi-square distributions

Usage

```
plotRMSEAdist(rmsea, n, df, ptile=NULL, caption=NULL, rmseaScale = TRUE, group=1)
```

Arguments

`rmsea`	The vector of RMSEA values to be plotted
`n`	Sample size of a dataset
`df`	Model degrees of freedom
`ptile`	The percentile rank of the distribution of the first RMSEA that users wish to plot a vertical line in the resulting graph
`caption`	The name vector of each element of `rmsea`

`rmseaScale` If `TRUE`, the RMSEA scale is used in the x-axis. If `FALSE`, the chi-square scale is used in the x-axis.

`group` The number of group that is used to calculate RMSEA.

Details

This function creates overlappling plots of the sampling distribution of RMSEA based on non-central chi-square distribution (MacCallum, Browne, & Suguwara, 1996). First, the noncentrality parameter (λ) is calculated from RMSEA (Steiger, 1998; Dudgeon, 2004) by

$$\lambda = (N-1)d\varepsilon^2/K,$$

where N is sample size, d is the model degree of freedom, K is the number of groupand ε is the population RMSEA. Next, the noncentral chi-square distribution with a specified degree of freedom and noncentrality parameter is plotted. Thus, the x-axis represent the sample chi-square value. The sample chi-square value can be transformed to the sample RMSEA scale ($\hat{\varepsilon}$) by

$$\hat{\varepsilon} = \sqrt{K}\sqrt{\frac{\chi^2 - d}{(N-1)d}},$$

where χ^2 is the chi-square value obtained from the noncentral chi-square distribution.

Author(s)

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

References

Dudgeon, P. (2004). A note on extending Steiger's (1998) multiple sample RMSEA adjustment to other noncentrality parameter-based statistic. *Structural Equation Modeling, 11*, 305-319.

MacCallum, R. C., Browne, M. W., & Sugawara, H. M. (1996). Power analysis and determination of sample size for covariance structure modeling. *Psychological Methods, 1,* 130-149.

Steiger, J. H. (1998). A note on multiple sample extensions of the RMSEA fit index. *Structural Equation Modeling, 5*, 411-419.

See Also

- `plotRMSEApower` to plot the statistical power based on population RMSEA given the sample size
- `findRMSEApower` to find the statistical power based on population RMSEA given a sample size
- `findRMSEAsamplesize` to find the minium sample size for a given statistical power based on population RMSEA

Examples

```
plotRMSEAdist(rmsea=c(.05, .08), n=200, df=20, ptile=0.95, rmseaScale = TRUE)
plotRMSEAdist(rmsea=c(.05, .01), n=200, df=20, ptile=0.05, rmseaScale = FALSE)
```

plotRMSEApower *Plot power curves for RMSEA*

Description

Plots power of RMSEA over a range of sample sizes

Usage

```
plotRMSEApower(rmsea0, rmseaA, df, nlow, nhigh, steps=1, alpha=.05, group=1)
```

Arguments

rmsea0	Null RMSEA
rmseaA	Alternative RMSEA
df	Model degrees of freedom
nlow	Lower sample size
nhigh	Upper sample size
steps	Increase in sample size for each iteration. Smaller values of steps will lead to more precise plots. However, smaller step sizes means a longer run time.
alpha	Alpha level used in power calculations
group	The number of group that is used to calculate RMSEA.

Details

This function creates plot of power for RMSEA against a range of sample sizes. The plot places sample size on the horizontal axis and power on the vertical axis. The user should indicate the lower and upper values for sample size and the sample size between each estimate ("step size") We strongly urge the user to read the sources below (see References) before proceeding. A web version of this function is available at: http://quantpsy.org/rmsea/rmseaplot.htm.

Value

1. plot Plot of power for RMSEA against a range of sample sizes

Author(s)

Alexander M. Schoemann (University of Kansas; <schoemann@ku.edu>) Kristopher J. Preacher (Vanderbilt University; <kris.preacher@vanderbilt.edu>) Donna L. Coffman (Pennsylvania State University; <dlc30@psu.edu.>)

References

MacCallum, R. C., Browne, M. W., & Cai, L. (2006). Testing differences between nested covariance structure models: Power analysis and null hypotheses. *Psychological Methods, 11,* 19-35.

MacCallum, R. C., Browne, M. W., & Sugawara, H. M. (1996). Power analysis and determination of sample size for covariance structure modeling. *Psychological Methods, 1,* 130-149.

MacCallum, R. C., Lee, T., & Browne, M. W. (2010). The issue of isopower in power analysis for tests of structural equation models. *Structural Equation Modeling, 17,* 23-41.

Preacher, K. J., Cai, L., & MacCallum, R. C. (2007). Alternatives to traditional model comparison strategies for covariance structure models. In T. D. Little, J. A. Bovaird, & N. A. Card (Eds.), *Modeling contextual effects in longitudinal studies* (pp. 33-62). Mahwah, NJ: Lawrence Erlbaum Associates.

Steiger, J. H. (1998). A note on multiple sample extensions of the RMSEA fit index. *Structural Equation Modeling, 5,* 411-419.

Steiger, J. H., & Lind, J. C. (1980, June). *Statistically based tests for the number of factors.* Paper presented at the annual meeting of the Psychometric Society, Iowa City, IA.

See Also

- `plotRMSEAdist` to visualize the RMSEA distributions
- `findRMSEApower` to find the statistical power based on population RMSEA given a sample size
- `findRMSEAsamplesize` to find the minium sample size for a given statistical power based on population RMSEA

Examples

```
plotRMSEApower(.025, .075, 23, 100, 500, 10)
```

`probe2WayMC` *Probing two-way interaction on the residual-centered latent interaction*

Description

Probing interaction for simple intercept and simple slope for the no-centered or mean-centered latent two-way interaction

Usage

```
probe2WayMC(fit, nameX, nameY, modVar, valProbe)
```

Arguments

`fit`	The lavaan model object used to evaluate model fit
`nameX`	The vector of the factor names used as the predictors. The first-order factor will be listed first. The last name must be the name representing the interaction term.
`nameY`	The name of factor that is used as the dependent variable.
`modVar`	The name of factor that is used as a moderator. The effect of the other independent factor on each moderator variable value will be probed.
`valProbe`	The values of the moderator that will be used to probe the effect of the other independent factor.

Details

Before using this function, researchers need to make the products of the indicators between the first-order factors using mean centering (Marsh, Wen, & Hau, 2004). Note that the double-mean centering may not be appropriate for probing interaction if researchers are interested in simple intercepts. The mean or double-mean centering can be done by the `indProd` function. The indicator products can be made for all possible combination or matched-pair approach (Marsh et al., 2004). Next, the hypothesized model with the regression with latent interaction will be used to fit all original indicators and the product terms. See the example for how to fit the product term below. Once the lavaan result is obtained, this function will be used to probe the interaction.

Let that the latent interaction model regressing the dependent variable (Y) on the independent varaible (X) and the moderator (Z) be

$$Y = b_0 + b_1X + b_2Z + b_3XZ + r,$$

where b_0 is the estimated intercept or the expected value of Y when both X and Z are 0, b_1 is the effect of X when Z is 0, b_2 is the effect of Z when X is 0, b_3 is the interaction effect between X and Z, and r is the residual term.

For probing two-way interaction, the simple intercept of the independent variable at each value of the moderator (Aiken & West, 1991; Cohen, Cohen, West, & Aiken, 2003; Preacher, Curran, & Bauer, 2006) can be obtained by

$$b_{0|X=0,Z} = b_0 + b_2Z.$$

The simple slope of the independent varaible at each value of the moderator can be obtained by

$$b_{X|Z} = b_1 + b_3Z.$$

The variance of the simple intercept formula is

$$Var\left(b_{0|X=0,Z}\right) = Var\left(b_0\right) + 2ZCov\left(b_0, b_2\right) + Z^2Var\left(b_2\right)$$

where Var denotes the variance of a parameter estimate and Cov denotes the covariance of two parameter estimates.

The variance of the simple slope formula is

$$Var\left(b_{X|Z}\right) = Var\left(b_1\right) + 2ZCov\left(b_1, b_3\right) + Z^2Var\left(b_3\right)$$

Wald statistic is used for test statistic.

Value

A list with two elements:

1. SimpleIntercept The intercepts given each value of the moderator. This element will be shown only if the factor intercept is estimated (e.g., not fixed as 0).
2. SimpleSlope The slopes given each value of the moderator.

In each element, the first column represents the values of the moderators specified in the `valProbe` argument. The second column is the simple intercept or simple slope. The third column is the standard error of the simple intercept or simple slope. The fourth column is the Wald (z) statistic. The fifth column is the p-value testing whether the simple intercepts or slopes are different from 0.

Author(s)

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

References

Aiken, L. S., & West, S. G. (1991). Multiple regression: Testing and interpreting interactions. Newbury Park, CA: Sage.

Cohen, J., Cohen, P., West, S. G., & Aiken, L. S. (2003). Applied multiple regression/correlation analysis for the behavioral sciences (3rd ed.). New York: Routledge.

Marsh, H. W., Wen, Z., & Hau, K. T. (2004). Structural equation models of latent interactions: Evaluation of alternative estimation strategies and indicator construction. *Psychological Methods, 9*, 275-300.

Preacher, K. J., Curran, P. J., & Bauer, D. J. (2006). Computational tools for probing interactions in multiple linear regression, multilevel modeling, and latent curve analysis. *Journal of Educational and Behavioral Statistics, 31*, 437-448.

See Also

- `indProd` For creating the indicator products with no centering, mean centering, double-mean centering, or residual centering.
- `probe3WayMC` For probing the three-way latent interaction when the results are obtained from mean-centering, or double-mean centering.
- `probe2WayRC` For probing the two-way latent interaction when the results are obtained from residual-centering approach.
- `probe3WayRC` For probing the two-way latent interaction when the results are obtained from residual-centering approach.
- `plotProbe` Plot the simple intercepts and slopes of the latent interaction.

Examples

```
library(lavaan)

dat2wayMC <- indProd(dat2way, 1:3, 4:6)

model1 <- "
f1 =~ x1 + x2 + x3
f2 =~ x4 + x5 + x6
f12 =~ x1.x4 + x2.x5 + x3.x6
f3 =~ x7 + x8 + x9
f3 ~ f1 + f2 + f12
f12 ~~0*f1
f12 ~~ 0*f2
x1 ~ 0*1
x4 ~ 0*1
x1.x4 ~ 0*1
x7 ~ 0*1
f1 ~ NA*1
f2 ~ NA*1
f12 ~ NA*1
f3 ~ NA*1
"

fitMC2way <- sem(model1, data=dat2wayMC, meanstructure=TRUE, std.lv=FALSE)
summary(fitMC2way)

result2wayMC <- probe2WayMC(fitMC2way, c("f1", "f2", "f12"), "f3", "f2", c(-1, 0, 1))
result2wayMC
```

probe2WayRC *Probing two-way interaction on the residual-centered latent interaction*

Description

Probing interaction for simple intercept and simple slope for the residual-centered latent two-way interaction (Pornprasertmanit, Schoemann, Geldhof, & Little, submitted)

Usage

```
probe2WayRC(fit, nameX, nameY, modVar, valProbe)
```

Arguments

fit	The lavaan model object used to evaluate model fit
nameX	The vector of the factor names used as the predictors. The first-order factor will be listed first. The last name must be the name representing the interaction term.
nameY	The name of factor that is used as the dependent variable.
modVar	The name of factor that is used as a moderator. The effect of the other independent factor on each moderator variable value will be probed.
valProbe	The values of the moderator that will be used to probe the effect of the other independent factor.

Details

Before using this function, researchers need to make the products of the indicators between the first-order factors and residualize the products by the original indicators (Lance, 1988; Little, Bovaird, & Widaman, 2006). The process can be automated by the indProd function. Note that the indicator products can be made for all possible combination or matched-pair approach (Marsh et al., 2004). Next, the hypothesized model with the regression with latent interaction will be used to fit all original indicators and the product terms. See the example for how to fit the product term below. Once the lavaan result is obtained, this function will be used to probe the interaction.

The probing process on residual-centered latent interaction is based on transforming the residual-centered result into the no-centered result. See Pornprasertmanit, Schoemann, Geldhof, and Little (submitted) for further details. Note that this approach based on a strong assumption that the first-order latent variables are normally distributed. The probing process is applied after the no-centered result (parameter estimates and their covariance matrix among parameter estimates) has been computed. See the probe2WayMC for further details.

Value

A list with two elements:

1. SimpleIntercept The intercepts given each value of the moderator. This element will be shown only if the factor intercept is estimated (e.g., not fixed as 0).
2. SimpleSlope The slopes given each value of the moderator.

In each element, the first column represents the values of the moderators specified in the valProbe argument. The second column is the simple intercept or simple slope. The third column is the standard error of the simple intercept or simple slope. The fourth column is the Wald (z) statistic. The fifth column is the p-value testing whether the simple intercepts or slopes are different from 0.

Author(s)

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

References

Lance, C. E. (1988). Residual centering, exploratory and confirmatory moderator analysis, and decomposition of effects in path models containing interactions. *Applied Psychological Measurement, 12*, 163-175.

Little, T. D., Bovaird, J. A., & Widaman, K. F. (2006). On the merits of orthogonalizing powered and product terms: Implications for modeling interactions. *Structural Equation Modeling, 13*, 497-519.

Marsh, H. W., Wen, Z., & Hau, K. T. (2004). Structural equation models of latent interactions: Evaluation of alternative estimation strategies and indicator construction. *Psychological Methods, 9*, 275-300.

Pornprasertmanit, S., Schoemann, A. M., Geldhof, G. J., & Little, T. D. (submitted). *Probing latent interaction estimated with a residual centering approach.*

See Also

- `indProd` For creating the indicator products with no centering, mean centering, double-mean centering, or residual centering.
- `probe2WayMC` For probing the two-way latent interaction when the results are obtained from mean-centering, or double-mean centering.
- `probe3WayMC` For probing the three-way latent interaction when the results are obtained from mean-centering, or double-mean centering.
- `probe3WayRC` For probing the two-way latent interaction when the results are obtained from residual-centering approach.
- `plotProbe` Plot the simple intercepts and slopes of the latent interaction.

Examples

```
library(lavaan)

dat2wayRC <- orthogonalize(dat2way, 1:3, 4:6)

model1 <- "
f1 =~ x1 + x2 + x3
f2 =~ x4 + x5 + x6
f12 =~ x1.x4 + x2.x5 + x3.x6
f3 =~ x7 + x8 + x9
f3 ~ f1 + f2 + f12
f12 ~~0*f1
f12 ~~ 0*f2
x1 ~ 0*1
x4 ~ 0*1
x1.x4 ~ 0*1
x7 ~ 0*1
f1 ~ NA*1
f2 ~ NA*1
f12 ~ NA*1
f3 ~ NA*1
"
```

```
fitRC2way <- sem(model1, data=dat2wayRC, meanstructure=TRUE, std.lv=FALSE)
summary(fitRC2way)

result2wayRC <- probe2WayRC(fitRC2way, c("f1", "f2", "f12"), "f3", "f2", c(-1, 0, 1))
result2wayRC
```

`probe3WayMC` *Probing two-way interaction on the residual-centered latent interaction*

Description

Probing interaction for simple intercept and simple slope for the no-centered or mean-centered latent two-way interaction

Usage

```
probe3WayMC(fit, nameX, nameY, modVar, valProbe1, valProbe2)
```

Arguments

`fit`	The lavaan model object used to evaluate model fit
`nameX`	The vector of the factor names used as the predictors. The three first-order factors will be listed first. Then the second-order factors will be listeed. The last element of the name will represent the three-way interaction. Note that the fourth element must be the interaction between the first and the second variables. The fifth element must be the interaction between the first and the third variables. The sixth element must be the interaction between the second and the third variables.
`nameY`	The name of factor that is used as the dependent variable.
`modVar`	The name of two factors that are used as the moderators. The effect of the independent factor on each combination of the moderator variable values will be probed.
`valProbe1`	The values of the first moderator that will be used to probe the effect of the independent factor.
`valProbe2`	The values of the second moderator that will be used to probe the effect of the independent factor.

Details

Before using this function, researchers need to make the products of the indicators between the first-order factors using mean centering (Marsh, Wen, & Hau, 2004). Note that the double-mean centering may not be appropriate for probing interaction if researchers are interested in simple intercepts. The mean or double-mean centering can be done by the `indProd` function. The indicator products can be made for all possible combination or matched-pair approach (Marsh et al., 2004). Next, the hypothesized model with the regression with latent interaction will be used to fit all original indicators and the product terms. See the example for how to fit the product term below. Once the lavaan result is obtained, this function will be used to probe the interaction.

Let that the latent interaction model regressing the dependent variable (Y) on the independent varaible (X) and two moderators (Z and W) be

$$Y = b_0 + b_1X + b_2Z + b_3W + b_4XZ + b_5XW + b_6ZW + b_7XZW + r,$$

where b_0 is the estimated intercept or the expected value of Y when X, Z, and W are 0, b_1 is the effect of X when Z and W are 0, b_2 is the effect of Z when X and W is 0, b_3 is the effect of W when X and Z are 0, b_4 is the interaction effect between X and Z when W is 0, b_5 is the interaction effect between X and W when Z is 0, b_6 is the interaction effect between Z and W when X is 0, b_7 is the three-way interaction effect between X, Z, and W, and r is the residual term.

For probing three-way interaction, the simple intercept of the independent variable at the specific values of the moderators (Aiken & West, 1991) can be obtained by

$$b_{0|X=0,Z,W} = b_0 + b_2Z + b_3W + b_6ZW.$$

The simple slope of the independent varaible at the specific values of the moderators can be obtained by

$$b_{X|Z,W} = b_1 + b_3Z + b_4W + b_7ZW.$$

The variance of the simple intercept formula is

$$Var\left(b_{0|X=0,Z,W}\right) = Var\left(b_0\right)+Z^2Var\left(b_2\right)+W^2Var\left(b_3\right)+Z^2W^2Var\left(b_6\right)+2ZCov\left(b_0,b_2\right)+2WCov\left(b_0,b_3\right)+2Z$$

where Var denotes the variance of a parameter estimate and Cov denotes the covariance of two parameter estimates.

The variance of the simple slope formula is

$$Var\left(b_{X|Z,W}\right) = Var\left(b_1\right)+Z^2Var\left(b_4\right)+W^2Var\left(b_5\right)+Z^2W^2Var\left(b_7\right)+2ZCov\left(b_1,b_4\right)+2WCov\left(b_1,b_5\right)+2ZWC$$

Wald statistic is used for test statistic.

Value

A list with two elements:

1. SimpleIntercept The intercepts given each value of the moderator. This element will be shown only if the factor intercept is estimated (e.g., not fixed as 0).
2. SimpleSlope The slopes given each value of the moderator.

In each element, the first column represents the values of the first moderator specified in the `valProbe1` argument. The second column represents the values of the second moderator specified in the `valProbe2` argument. The third column is the simple intercept or simple slope. The fourth column is the standard error of the simple intercept or simple slope. The fifth column is the Wald (z) statistic. The sixth column is the p-value testing whether the simple intercepts or slopes are different from 0.

Author(s)

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

References

Aiken, L. S., & West, S. G. (1991). Multiple regression: Testing and interpreting interactions. Newbury Park, CA: Sage.

Marsh, H. W., Wen, Z., & Hau, K. T. (2004). Structural equation models of latent interactions: Evaluation of alternative estimation strategies and indicator construction. *Psychological Methods, 9*, 275-300.

See Also

- indProd For creating the indicator products with no centering, mean centering, double-mean centering, or residual centering.
- probe2WayMC For probing the two-way latent interaction when the results are obtained from mean-centering, or double-mean centering.
- probe2WayRC For probing the two-way latent interaction when the results are obtained from residual-centering approach.
- probe3WayRC For probing the two-way latent interaction when the results are obtained from residual-centering approach.
- plotProbe Plot the simple intercepts and slopes of the latent interaction.

Examples

```
library(lavaan)

dat3wayMC <- indProd(dat3way, 1:3, 4:6, 7:9)

model3 <- "
f1 =~ x1 + x2 + x3
f2 =~ x4 + x5 + x6
f3 =~ x7 + x8 + x9
f12 =~ x1.x4 + x2.x5 + x3.x6
f13 =~ x1.x7 + x2.x8 + x3.x9
f23 =~ x4.x7 + x5.x8 + x6.x9
f123 =~ x1.x4.x7 + x2.x5.x8 + x3.x6.x9
f4 =~ x10 + x11 + x12
f4 ~ f1 + f2 + f3 + f12 + f13 + f23 + f123
f1 ~~ 0*f12
f1 ~~ 0*f13
f1 ~~ 0*f123
f2 ~~ 0*f12
f2 ~~ 0*f23
f2 ~~ 0*f123
f3 ~~ 0*f13
f3 ~~ 0*f23
f3 ~~ 0*f123
f12 ~~ 0*f123
f13 ~~ 0*f123
f23 ~~ 0*f123
x1 ~ 0*1
x4 ~ 0*1
x7 ~ 0*1
x10 ~ 0*1
x1.x4 ~ 0*1
x1.x7 ~ 0*1
x4.x7 ~ 0*1
x1.x4.x7 ~ 0*1
f1 ~ NA*1
f2 ~ NA*1
f3 ~ NA*1
f12 ~ NA*1
f13 ~ NA*1
f23 ~ NA*1
f123 ~ NA*1
```

```
f4 ~ NA*1
"

fitMC3way <- sem(model3, data=dat3wayMC, meanstructure=TRUE, std.lv=FALSE)
summary(fitMC3way)

result3wayMC <- probe3WayMC(fitMC3way, c("f1", "f2", "f3", "f12", "f13", "f23", "f123"), "f4", c("f1", "f2"
result3wayMC
```

`probe3WayRC` *Probing three-way interaction on the residual-centered latent interaction*

Description

Probing interaction for simple intercept and simple slope for the residual-centered latent three-way interaction (Pornprasertmanit, Schoemann, Geldhof, & Little, submitted)

Usage

```
probe3WayRC(fit, nameX, nameY, modVar, valProbe1, valProbe2)
```

Arguments

`fit`	The lavaan model object used to evaluate model fit
`nameX`	The vector of the factor names used as the predictors. The three first-order factors will be listed first. Then the second-order factors will be listeed. The last element of the name will represent the three-way interaction. Note that the fourth element must be the interaction between the first and the second variables. The fifth element must be the interaction between the first and the third variables. The sixth element must be the interaction between the second and the third variables.
`nameY`	The name of factor that is used as the dependent variable.
`modVar`	The name of two factors that are used as the moderators. The effect of the independent factor on each combination of the moderator variable values will be probed.
`valProbe1`	The values of the first moderator that will be used to probe the effect of the independent factor.
`valProbe2`	The values of the second moderator that will be used to probe the effect of the independent factor.

Details

Before using this function, researchers need to make the products of the indicators between the first-order factors and residualize the products by the original indicators (Lance, 1988; Little, Bovaird, & Widaman, 2006). The process can be automated by the `indProd` function. Note that the indicator products can be made for all possible combination or matched-pair approach (Marsh et al., 2004). Next, the hypothesized model with the regression with latent interaction will be used to fit all original indicators and the product terms (Geldhof, Pornprasertmanit, Schoemann, & Little, in press). See the example for how to fit the product term below. Once the lavaan result is obtained, this function will be used to probe the interaction.

The probing process on residual-centered latent interaction is based on transforming the residual-centered result into the no-centered result. See Pornprasertmanit, Schoemann, Geldhof, and Little (submitted) for further details. Note that this approach based on a strong assumption that the first-order latent variables are normally distributed. The probing process is applied after the no-centered result (parameter estimates and their covariance matrix among parameter estimates) has been computed See the `probe3WayMC` for further details.

Value

A list with two elements:

1. SimpleIntercept The intercepts given each value of the moderator. This element will be shown only if the factor intercept is estimated (e.g., not fixed as 0).
2. SimpleSlope The slopes given each value of the moderator.

In each element, the first column represents the values of the first moderator specified in the `valProbe1` argument. The second column represents the values of the second moderator specified in the `valProbe2` argument. The third column is the simple intercept or simple slope. The fourth column is the standard error of the simple intercept or simple slope. The fifth column is the Wald (z) statistic. The sixth column is the p-value testing whether the simple intercepts or slopes are different from 0.

Author(s)

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

References

Geldhof, G. J., Pornprasertmanit, S., Schoemann, A., & Little, T. D. (in press). Orthogonalizing through residual centering: Applications and caveats. *Educational and Psychological Measurement.*

Lance, C. E. (1988). Residual centering, exploratory and confirmatory moderator analysis, and decomposition of effects in path models containing interactions. *Applied Psychological Measurement, 12*, 163-175.

Little, T. D., Bovaird, J. A., & Widaman, K. F. (2006). On the merits of orthogonalizing powered and product terms: Implications for modeling interactions. *Structural Equation Modeling, 13*, 497-519.

Marsh, H. W., Wen, Z., & Hau, K. T. (2004). Structural equation models of latent interactions: Evaluation of alternative estimation strategies and indicator construction. *Psychological Methods, 9*, 275-300.

Pornprasertmanit, S., Schoemann, A. M., Geldhof, G. J., & Little, T. D. (submitted). *Probing latent interaction estimated with a residual centering approach.*

See Also

- `indProd` For creating the indicator products with no centering, mean centering, double-mean centering, or residual centering.
- `probe2WayMC` For probing the two-way latent interaction when the results are obtained from mean-centering, or double-mean centering.
- `probe3WayMC` For probing the three-way latent interaction when the results are obtained from mean-centering, or double-mean centering.

- probe2WayRC For probing the two-way latent interaction when the results are obtained from residual-centering approach.
- plotProbe Plot the simple intercepts and slopes of the latent interaction.

Examples

```
library(lavaan)

dat3wayRC <- orthogonalize(dat3way, 1:3, 4:6, 7:9)

model3 <- "
f1 =~ x1 + x2 + x3
f2 =~ x4 + x5 + x6
f3 =~ x7 + x8 + x9
f12 =~ x1.x4 + x2.x5 + x3.x6
f13 =~ x1.x7 + x2.x8 + x3.x9
f23 =~ x4.x7 + x5.x8 + x6.x9
f123 =~ x1.x4.x7 + x2.x5.x8 + x3.x6.x9
f4 =~ x10 + x11 + x12
f4 ~ f1 + f2 + f3 + f12 + f13 + f23 + f123
f1 ~~ 0*f12
f1 ~~ 0*f13
f1 ~~ 0*f123
f2 ~~ 0*f12
f2 ~~ 0*f23
f2 ~~ 0*f123
f3 ~~ 0*f13
f3 ~~ 0*f23
f3 ~~ 0*f123
f12 ~~ 0*f123
f13 ~~ 0*f123
f23 ~~ 0*f123
x1 ~ 0*1
x4 ~ 0*1
x7 ~ 0*1
x10 ~ 0*1
x1.x4 ~ 0*1
x1.x7 ~ 0*1
x4.x7 ~ 0*1
x1.x4.x7 ~ 0*1
f1 ~ NA*1
f2 ~ NA*1
f3 ~ NA*1
f12 ~ NA*1
f13 ~ NA*1
f23 ~ NA*1
f123 ~ NA*1
f4 ~ NA*1
"

fitRC3way <- sem(model3, data=dat3wayRC, meanstructure=TRUE, std.lv=FALSE)
summary(fitRC3way)

result3wayRC <- probe3WayRC(fitRC3way, c("f1", "f2", "f3", "f12", "f13", "f23", "f123"), "f4", c("f1", "f2"
result3wayRC
```

`residualCovariate` *Residual centered all target indicators by covariates*

Description

This function will regress target variables on the covariate and replace the target variables by the residual of the regression analysis. This procedure is useful to control the covariate from the analysis model (Geldhof, Pornprasertmanit, Schoemann, & Little, in press).

Usage

```
residualCovariate(data, targetVar, covVar)
```

Arguments

`data`	The desired data to be transformed.
`targetVar`	Varible names or the position of indicators that users wish to be residual centered (as dependent variables)
`covVar`	Covariate names or the position of the covariates using for residual centering (as independent variables) onto target variables

Value

The data that the target variables replaced by the residuals

Author(s)

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

References

Geldhof, G. J., Pornprasertmanit, S., Schoemann, A. M., & Little, T. D. (in press). Orthogonalizing through residual centering: Applications and caveats. *Educational and Psychological Measurement*.

See Also

`indProd` For creating the indicator products with no centering, mean centering, double-mean centering, or residual centering.

Examples

```
dat <- residualCovariate(attitude, 2:7, 1)
```

runMI *Multiply impute and analyze data using lavaan*

Description

This function takes data with missing observations, multiple imputes the data, runs a SEM using lavaan and combines the results using Rubin's rules. Note that parmeter estimates and standard errors are pooled by the Rubin's (1987) rule. The chi-square statistics and the related fit indices are pooled by the method described in "chi" argument. SRMR is calculated based on the average model-implied means and covariance matrices across imputations.

Usage

```
runMI(model, data, m, miArgs=list(), chi="all", miPackage="Amelia",
seed=12345, fun, ...)
cfa.mi(model, data, m, miArgs=list(), miPackage="Amelia", chi="all",
seed=12345, ...)
sem.mi(model, data, m, miArgs=list(), miPackage="Amelia", chi="all",
seed=12345, ...)
growth.mi(model, data, m, miArgs=list(), miPackage="Amelia", chi="all",
seed=12345, ...)
lavaan.mi(model, data, m, miArgs=list(), miPackage="Amelia", chi="all",
seed=12345, ...)
```

Arguments

model	lavaan syntax for the the model to be analyzed.
data	Data frame with missing observations or a list of data frames where each data frame is one imputed data set (for imputed data generated outside of the function). If a list of data frames is supplied, then other options can be left at the default.
m	Number of imputations wanted.
miArgs	Addition arguments for the multiple-imputation function. The arguments should be put in a list (see example below).
miPackage	Package to be used for imputation. Currently these functions only support "Amelia" or "mice" for imputation.
chi	The method to combine the chi-square. Can be one of the following: "MR" for the method proposed for Meng & Rubin (1992), "Mplus" for the method used in Mplus (Asparouhov & Muthen, 2010), "LMRR" for the method proposed by Li, Meng, Raghunathan, & Rubin (1991), and "all" to show the three methods in the output. The default is "all".
seed	Random number seed to be used in imputations.
fun	The character of the function name used in running lavaan model ("cfa", "sem", "growth", "lavaan").
...	Other arguments to be passed to the specified lavaan function ("cfa", "sem", "growth", "lavaan").

Value

The lavaanStar object which contains the original lavaan object (where the appropriate parameter estimates, appropriate standard errors, and chi-squares are filled), the additional fit-index values of the null model, which need to be adjusted to multiple datasets, and the information from pooling multiple results.

Author(s)

Alexander M. Schoemann (University of Kansas; <schoemann@ku.edu>) Patrick Miller (University of Kansas; <patr1ckm@ku.edu>) Sunthud Pornprasertmanit (University of Kansas; <psunthud@ku.edu>) Mijke Rhemtulla (University of Kansas; <mijke@ku.edu>) Alexander Robitzsch (Federal Institute for Education Research, Innovation, and Development of the Austrian School System, Salzburg, Austria; <a.robitzsch@bifie.at>) Craig Enders (Arizona State University; <Craig.Enders@asu.edu>) Mauricio Garnier Villarreal (University of Kansas; <mgv@ku.edu>) Yves Rosseel (Ghent University; <Yves.Rosseel@UGent.be>)

References

Asparouhov T. & Muthen B. (2010).*Chi-Square Statistics with Multiple Imputation*. Technical Report. www.statmodel.com.

Li, K.H., Meng, X.-L., Raghunathan, T.E. and Rubin, D.B. (1991). Significance Levels From Repeated p-values with Multiply-Imputed Data. *Statistica Sinica, 1*, 65-92.

Meng, X.L. & Rubin, D.B. (1992). Performing likelihood ratio tests with multiply-imputed data sets. *Biometrika, 79*, 103 - 111.

Rubin, D.B. (1987) *Multiple Imputation for Nonresponse in Surveys.* J. Wiley & Sons, New York.

Examples

```
library(lavaan)

HS.model <- ' visual  =~ x1 + x2 + x3
              textual =~ x4 + x5 + x6
              speed   =~ x7 + x8 + x9 '

HSMiss <- HolzingerSwineford1939[,paste("x", 1:9, sep="")]
randomMiss <- rbinom(prod(dim(HSMiss)), 1, 0.1)
randomMiss <- matrix(as.logical(randomMiss), nrow=nrow(HSMiss))
HSMiss[randomMiss] <- NA

out <- cfa.mi(HS.model, data=HSMiss, m = 3, chi="all")
summary(out)
inspect(out, "fit")
inspect(out, "impute")

## Not run:
##Multiple group example
HSMiss2 <- cbind(HSMiss, school = HolzingerSwineford1939[,"school"])
out2 <- cfa.mi(HS.model, data=HSMiss2, m = 3, miArgs=list(noms="school"), chi="MR", group="school")
summary(out2)
inspect(out2, "fit")
inspect(out2, "impute")

##Example using previously imputed data with runMI
library(Amelia)
```

```
modsim <- '
f1 =~ 0.7*y1+0.7*y2+0.7*y3
f2 =~ 0.7*y4+0.7*y5+0.7*y6
f3 =~ 0.7*y7+0.7*y8+0.7*y9'

mod <- '
f1 =~ y1+y2+y3
f2 =~ y4+y5+y6
f3 =~ y7+y8+y9'

datsim <- simulateData(modsim,model.type="cfa", meanstructure=TRUE,
std.lv=TRUE, sample.nobs=c(200,200))
randomMiss2 <- rbinom(prod(dim(datsim)), 1, 0.1)
randomMiss2 <- matrix(as.logical(randomMiss2), nrow=nrow(datsim))
datsim[randomMiss2] <- NA
datsimMI <- amelia(datsim,m=3, noms="group")

out3 <- runMI(mod, data=datsimMI$imputations, chi="LMRR", group="group", fun="cfa")
summary(out3)
inspect(out3, "fit")
inspect(out3, "impute")

## End(Not run)
```

`simParcel` *Simulated Data set to Demonstrate Random Allocations of Parcels*

Description

A simulated data set with 2 factors with 9 indicators for each factor

Usage

```
data(simParcel)
```

Format

A data frame with 800 observations of 18 variables.

f1item1 Item 1 loading on factor 1

f1item2 Item 2 loading on factor 1

f1item3 Item 3 loading on factor 1

f1item4 Item 4 loading on factor 1

f1item5 Item 5 loading on factor 1

f1item6 Item 6 loading on factor 1

f1item7 Item 7 loading on factor 1

f1item8 Item 8 loading on factor 1

f1item9 Item 9 loading on factor 1

f2item1 Item 1 loading on factor 2

f2item2 Item 2 loading on factor 2
f2item3 Item 3 loading on factor 2
f2item4 Item 4 loading on factor 2
f2item5 Item 5 loading on factor 2
f2item6 Item 6 loading on factor 2
f2item7 Item 7 loading on factor 2
f2item8 Item 8 loading on factor 2
f2item9 Item 9 loading on factor 2

Source

Data was generated using the `simsem` package.

Examples

```
head(simParcel)
```

`skew` *Finding skewness*

Description

Finding skewness (g1) of an object

Usage

```
skew(object, population=FALSE)
```

Arguments

`object`	A vector used to find a skewness
`population`	`TRUE` to compute the parameter formula. `FALSE` to compute the sample statistic formula.

Details

The skewness computed is g1. The parameter skewness γ_2 formula is

$$\gamma_2 = \frac{\mu_3}{\mu_2^{3/2}},$$

where μ_i denotes the i order central moment.

The excessive kurtosis formula for sample statistic g_2 is

$$g_2 = \frac{k_3}{k_2^2},$$

where k_i are the i order *k*-statistic.

The standard error of the skewness is

$$Var(\hat{g}_2) = \frac{6}{N}$$

where N is the sample size.

Value

A value of a skewness with a test statistic if the population is specified as FALSE

Author(s)

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

References

Weisstein, Eric W. (n.d.). *Skewness.* Retrived from MathWorld–A Wolfram Web Resource http://mathworld.wolfram.com/Skewness.html

See Also

- kurtosis Find the univariate excessive kurtosis of a variable
- mardiaSkew Find the Mardia's multivariate skewness of a set of variables
- mardiaKurtosis Find the Mardia's multivariate kurtosis of a set of variables

Examples

```
skew(1:5)
```

---

splitSample *Randomly Split a Data Set into Halves*

Description

This function randomly splits a data set into two halves, and saves the resulting data sets to the same folder as the original.

Usage

```
splitSample(dataset,path="default", div=2, type="default", name="splitSample")
```

Arguments

dataset	The original data set to be divided. Can be a file path to a .csv or .dat file (headers will automatically be detected) or an R object (matrix or dataframe). (Windows users: file path must be specified using FORWARD SLASHES ONLY.)
path	File path to folder for output data sets. NOT REQUIRED if dataset is a filename. Specify ONLY if dataset is an R object, or desired output folder is not that of original data set. If path is specified as "object", output data sets will be returned as a list, and not saved to hard drive.
div	Number of output data sets. NOT REQUIRED if default, 2 halves.
type	Output file format ("dat" or "csv"). NOT REQUIRED unless desired output formatting differs from that of input, or dataset is an R object and csv formatting is desired.
name	Output file name. NOT REQUIRED unless desired output name differs from that of input, or input dataset is an R object. (If input is an R object and name is not specified, name will be "splitSample".)

Details

This function randomly orders the rows of a data set, divides the data set into two halves, and saves the halves to the same folder as the original data set, preserving the original formatting. Data set type (.csv or .dat) and formatting (headers) are automatically detected, and output data sets will preserve input type and formatting unless specified otherwise. Input can be in the form of a file path (.dat or .csv), or an R object (matrix or dataframe). If input is an R object and path is default, output data sets will be returned as a list object.

Value

dataL	List of output data sets. ONLY IF dataset is an R object and path is default. Otherwise, output will saved to hard drive with the same formatting as input.

Author(s)

Corbin Quick (University of Kansas; <corbinq@ku.edu>)

Examples

```
#### Input is .dat file
#splitSample("C:/Users/Default/Desktop/MYDATA.dat")
#### Output saved to "C:/Users/Default/Desktop/" in .dat format
#### Names are "MYDATA_s1.dat" and "MYDATA_s2.dat"

#### Input is R object
##Split CO2 dataset from the datasets package
library(datasets)
splitMyData <- splitSample(CO2, path="object")
summary(splitMyData[[1]])
summary(splitMyData[[2]])
#### Output object splitMyData becomes list of output data sets

#### Input is .dat file in "C:/" folder
#splitSample("C:/testdata.dat", path = "C:/Users/Default/Desktop/", type = "csv")
#### Output saved to "C:/Users/Default/Desktop/" in .csv format
#### Names are "testdata_s1.csv" and "testdata_s2.csv"

#### Input is R object
#splitSample(myData, path = "C:/Users/Default/Desktop/", name = "splitdata")
#### Output saved to "C:/Users/Default/Desktop/" in .dat format
#### Names are "splitdata_s1.dat" and "splitdata_s2.dat"
```

Index