Package 'lavaan.mi'

July 22, 2025

Encoding UTF-8

Version 0.1-0

Title Fit Structural Equation Models to Multiply Imputed Data

Description The primary purpose of 'lavaan.mi' is to extend the functionality of the R package 'lavaan', which implements structural equation modeling (SEM). When incomplete data have been multiply imputed, the imputed data sets can be analyzed by 'lavaan' using complete-data estimation methods, but results must be pooled across imputations (Rubin, 1987, <doi:10.1002/9780470316696>). The 'lavaan.mi' package automates the pooling of point and standard-error estimates, as well as a variety of test statistics, using a familiar interface that allows users to fit an SEM to multiple imputations as they would to a single data set using the 'lavaan' package.

Depends R(>= 4.0), lavaan(>= 0.6-18), methods

Imports stats, utils

Suggests Amelia, MASS, mice, parallel, testthat

License GPL (>= 2)

LazyData yes

LazyLoad yes

URL https://github.com/TDJorgensen/lavaan.mi

BugReports https://github.com/TDJorgensen/lavaan.mi/issues

Date 2025-03-07

RoxygenNote 7.3.2

NeedsCompilation no

Author Terrence D. Jorgensen [aut, cre] (ORCID: <https://orcid.org/0000-0001-5111-6773>), Yves Rosseel [ctb] (ORCID: <https://orcid.org/0000-0002-4129-4477>)

Maintainer Terrence D. Jorgensen <TJorgensen314@gmail.com>

Repository CRAN

Date/Publication 2025-03-10 14:30:02 UTC

Contents

`binHS5imps` *List of imputed Holzinger & Swineford (1939) dichotomized data*

Description

A version of the classic Holzinger and Swineford (1939) dataset, with missing data imposed on variables `x5` and `x9`:

Details

- `x5` is missing not at random (MNAR) by deleting the lowest 30% of `x5` values.
- `x9` is missing at random (MAR) conditional on age, by deleting `x9` values for the youngest 30% of subjects in the data.

The data are then dichotomized using a median split, and imputed 5 times using the syntax shown in the example. The data include only the 9 tests (`x1` through `x9`) and school.

Author(s)

Terrence D. Jorgensen (University of Amsterdam; `<TJorgensen314@gmail.com>`)

Source

The lavaan package.

References

Holzinger, K., & Swineford, F. (1939). *A study in factor analysis: The stability of a bifactor solution*. Supplementary Educational Monograph, no. 48. Chicago, IL: University of Chicago Press.

See Also

lavaan::HolzingerSwineford1939

Examples

```
data(HolzingerSwineford1939, package = "lavaan")

## impose missing data for example
HSMiss <- HolzingerSwineford1939[ , c(paste("x", 1:9, sep = ""),
                                      "ageyr","agemo","school")]
set.seed(123)
HSMiss$x5 <- ifelse(HSMiss$x5 <= quantile(HSMiss$x5, .3), NA, HSMiss$x5)
age <- HSMiss$ageyr + HSMiss$agemo/12
HSMiss$x9 <- ifelse(age <= quantile(age, .3), NA, HSMiss$x9)

## median split
HSbinary <- as.data.frame( lapply(HSMiss[ , paste0("x", 1:9)],
                                  FUN = cut, breaks = 2, labels = FALSE) )
HSbinary$school <- HSMiss$school

## impute binary missing data using mice package
library(mice)
set.seed(456)
miceImps <- mice(HSbinary)
## save imputations in a list of data.frames
binHS5imps <- list()
for (i in 1:miceImps$m) binHS5imps[[i]] <- complete(miceImps, action = i)
```

`calculate.D2` *Calculate the "D2" statistic*

Description

This is a utility function used to calculate the "D2" statistic for pooling test statistics across multiple imputations. This function is called by several functions used for lavaan.mi objects, such as `lavTestLRT.mi()`, `lavTestWald.mi()`, and `lavTestScore.mi()`. But this function can be used for any general scenario because it only requires a vector of χ^2 statistics (one from each imputation) and the degrees of freedom for the test statistic. See Li, Meng, Raghunathan, & Rubin (1991) and Enders (2010, chapter 8) for details about how it is calculated.

Usage

```
calculate.D2(w, DF = 0L, asymptotic = FALSE)
```

Arguments

w	numeric vector of Wald χ^2 statistics. Can also be Wald z statistics, which will be internally squared to make χ^2 statistics with one *df* (must set DF = 0L).
DF	degrees of freedom (*df*) of the χ^2 statistics. If DF = 0L (default), w is assumed to contain z statistics, which will be internally squared.
asymptotic	logical. If FALSE (default), the pooled test will be returned as an F-distributed statistic with numerator (df1) and denominator (df2) degrees of freedom. If TRUE, the pooled F statistic will be multiplied by its df1 on the assumption that its df2 is sufficiently large enough that the statistic will be asymptotically χ^2 distributed with df1.

Value

A numeric vector containing the test statistic, *df*, its p value, and 2 missing-data diagnostics: the relative invrease in variance (RIV, or average for multiparameter tests: ARIV) and the fraction missing information (FMI = ARIV / (1 + ARIV)).

Author(s)

Terrence D. Jorgensen (University of Amsterdam; <TJorgensen314@gmail.com>)

References

Enders, C. K. (2010). *Applied missing data analysis*. New York, NY: Guilford.

Li, K.-H., Meng, X.-L., Raghunathan, T. E., & Rubin, D. B. (1991). Significance levels from repeated p-values with multiply-imputed data. *Statistica Sinica, 1*(1), 65–92. Retrieved from https://www.jstor.org/stable/24303994

See Also

lavTestLRT.mi(), lavTestWald.mi(), lavTestScore.mi()

Examples

```
## generate a vector of chi-squared values, just for example
DF <- 3 # degrees of freedom
M <- 20 # number of imputations
CHI <- rchisq(M, DF)

## pool the "results"
calculate.D2(CHI, DF) # by default, an F statistic is returned
calculate.D2(CHI, DF, asymptotic = TRUE) # asymptotically chi-squared

## generate standard-normal values, for an example of Wald z tests
Z <- rnorm(M)
calculate.D2(Z) # default DF = 0 will square Z to make chisq(DF = 1)
## F test is equivalent to a t test with the denominator DF
```

HS20imps *List of imputed Holzinger & Swineford (1939) datasets*

Description

A version of the classic Holzinger and Swineford (1939) dataset, with missing data imposed on variables x5 and x9:

Details

- x5 is missing not at random (MNAR) by deleting the lowest 30% of x5 values.
- x9 is missing at random (MAR) conditional on age, by deleting x5 values for the youngest 30% of subjects in the data.

The data are imputed 20 times using the syntax shown in the example. The data include only age and school variables, along with 9 tests (x1 through x9).

Author(s)

Terrence D. Jorgensen (University of Amsterdam; <TJorgensen314@gmail.com>)

Source

The lavaan package.

References

Holzinger, K., & Swineford, F. (1939). *A study in factor analysis: The stability of a bifactor solution*. Supplementary Educational Monograph, no. 48. Chicago, IL: University of Chicago Press.

See Also

lavaan::HolzingerSwineford1939

Examples

```
data(HolzingerSwineford1939, package = "lavaan")

## impose missing data for example
HSMiss <- HolzingerSwineford1939[ , c(paste("x", 1:9, sep = ""),
                                      "ageyr","agemo","school")]
set.seed(123)
HSMiss$x5 <- ifelse(HSMiss$x5 <= quantile(HSMiss$x5, .3), NA, HSMiss$x5)
age <- HSMiss$ageyr + HSMiss$agemo/12
HSMiss$x9 <- ifelse(age <= quantile(age, .3), NA, HSMiss$x9)

## impute missing data with Amelia
library(Amelia)
```

```
set.seed(456)
HS.amelia <- amelia(HSMiss, m = 20, noms = "school", p2s = FALSE)
HS20imps <- HS.amelia$imputations
```

`lavaan.mi` *Fit a lavaan Model to Multiple Imputed Data Sets*

Description

This function fits a lavaan model to a list of imputed data sets.

Usage

```
lavaan.mi(model, data, ...)

cfa.mi(model, data, ...)

sem.mi(model, data, ...)

growth.mi(model, data, ...)
```

Arguments

`model`	The analysis model can be specified using lavaan `lavaan::model.syntax()` or a parameter table (as generated by `lavaan::lavaanify()` or returned by `lavaan::parTable()`).
`data`	A a `list` of imputed data sets, or an object class from which imputed data can be extracted. Recognized classes are `lavaan.mi` (stored in the `@DataList` slot), `amelia` (created by the Amelia package), or `mids` (created by the mice package).
`...`	additional arguments to pass to `lavaan::lavaan()` or `lavaan::lavaanList()`. See also `lavaan::lavOptions()`. Note that `lavaanList` provides parallel computing options, as well as a `FUN=` argument so the user can extract custom output after the model is fitted to each imputed data set (see **Examples**). TIP: If a custom `FUN` is used *and* `parallel = "snow"` is requested, the user-supplied function should explicitly call `library` or use `::` for any functions not part of the base distribution.

Value

A lavaan.mi object

Note

This functionality was originally provided via `runMI()` in the `semTools` package, but there are differences. See the README file on the GitHub page for this package (find link in DESCRIPTION).

Author(s)

Terrence D. Jorgensen (University of Amsterdam; <TJorgensen314@gmail.com>)

References

Enders, C. K. (2010). *Applied missing data analysis*. New York, NY: Guilford.

Rubin, D. B. (1987). *Multiple imputation for nonresponse in surveys*. New York, NY: Wiley. doi:10.1002/9780470316696

See Also

`poolSat()` for a more efficient method to obtain SEM results for multiple imputations

Examples

```
data(HS20imps) # import a list of 20 imputed data sets

## specify CFA model from lavaan's ?cfa help page
HS.model <- '
  visual  =~ x1 + x2 + x3
  textual =~ x4 + x5 + x6
  speed   =~ x7 + x8 + x9
'

## fit model to imputed data sets
fit <- cfa.mi(HS.model, data = HS20imps)


summary(fit, fit.measures = TRUE, fmi = TRUE)
summary(fit, standardized = "std.all", rsquare = TRUE)


## You can pass other lavaanList() arguments, such as FUN=, which allows
## you to save any custom output from each imputation's fitted model.

## An example with ordered-categorical data:
data(binHS5imps) # import a list of 5 imputed data sets

## Define a function to save a list with 2 custom outputs per imputation:
##  - zero-cell tables
##  - the obsolete "WRMR" fit index
myCustomFunc <- function(object) {
  list(wrmr      = lavaan::fitMeasures(object, "wrmr"),
       zeroCells = lavaan::lavInspect(object, "zero.cell.tables"))
}
## fit the model
catout <- cfa.mi(HS.model, data = binHS5imps, ordered = TRUE,
                 FUN = myCustomFunc)
## pooled results

summary(catout)
```

```
## extract custom output (per imputation)
sapply(catout@funList, function(x) x$wrmr) # WRMR for each imputation
catout@funList[[1]]$zeroCells # zero-cell tables for first imputation
catout@funList[[2]]$zeroCells # zero-cell tables for second imputation ...
```

`lavaan.mi-class` *Class for a lavaan Model Fitted to Multiple Imputations*

Description

This class extends the lavaan::lavaanList class, created by fitting a lavaan model to a list of data sets. In this case, the list of data sets are multiple imputations of missing data.

Usage

```
## S4 method for signature 'lavaan.mi'
show(object)

## S4 method for signature 'lavaan.mi'
summary(
  object,
  header = TRUE,
  fit.measures = FALSE,
 fm.args = list(standard.test = "default", scaled.test = "default", rmsea.ci.level =
    0.9, rmsea.h0.closefit = 0.05, rmsea.h0.notclosefit = 0.08, robust = TRUE,
    cat.check.pd = TRUE),
  estimates = TRUE,
  ci = FALSE,
  standardized = FALSE,
  std = standardized,
  cov.std = TRUE,
  rsquare = FALSE,
  fmi = FALSE,
  asymptotic = FALSE,
  scale.W = !asymptotic,
  omit.imps = c("no.conv", "no.se"),
  remove.unused = TRUE,
  modindices = FALSE,
  nd = 3L,
  ...
)

## S4 method for signature 'lavaan.mi'
nobs(object, total = TRUE)
```

```
## S4 method for signature 'lavaan.mi'
coef(object, type = "free", labels = TRUE, omit.imps = c("no.conv", "no.se"))

## S4 method for signature 'lavaan.mi'
vcov(
  object,
  type = c("pooled", "between", "within", "ariv"),
  scale.W = TRUE,
  omit.imps = c("no.conv", "no.se")
)

## S4 method for signature 'lavaan.mi'
fitted(object, omit.imps = c("no.conv", "no.se"))

## S4 method for signature 'lavaan.mi'
fitted.values(object, omit.imps = c("no.conv", "no.se"))

## S4 method for signature 'lavaan.mi'
fitMeasures(
  object,
  fit.measures = "all",
  baseline.model = NULL,
  h1.model = NULL,
 fm.args = list(standard.test = "default", scaled.test = "default", rmsea.ci.level =
    0.9, rmsea.h0.closefit = 0.05, rmsea.h0.notclosefit = 0.08, robust = 0.08,
    cat.check.pd = TRUE),
  output = "vector",
  omit.imps = c("no.conv", "no.se"),
  ...
)

## S4 method for signature 'lavaan.mi'
fitmeasures(
  object,
  fit.measures = "all",
  baseline.model = NULL,
  h1.model = NULL,
 fm.args = list(standard.test = "default", scaled.test = "default", rmsea.ci.level =
    0.9, rmsea.h0.closefit = 0.05, rmsea.h0.notclosefit = 0.08, robust = 0.08,
    cat.check.pd = TRUE),
  output = "vector",
  omit.imps = c("no.conv", "no.se"),
  ...
)
```

Arguments

object	An object of class lavaan.mi

`header`, `fit.measures`, `fm.args`, `estimates`, `ci`, `standardized`, `std`, `cov.std`, `rsquare`, `remove.unused`, `modindices`, `nd`, `output`
: See descriptions of `summary()` arguments in the help page for lavaan::lavaan class. Also see `lavaan::fitMeasures()` for arguments `fit.measures` and `fm.args`.

`fmi`
: `logical` indicating whether to add the Fraction Missing Information (FMI) and (average) relative increase in variance (ARIV) to the output.

`asymptotic`
: `logical`. If `FALSE` (typically a default, but see **Value** section for details using various methods), pooled tests (of fit or pooled estimates) will be *F* or *t* statistics with associated degrees of freedom (*df*). If `TRUE`, the (denominator) *df* are assumed to be sufficiently large for a *t* statistic to follow a normal distribution, so it is printed as a *z* statistic; likewise, *F* times its numerator *df* is printed, assumed to follow a χ^2 distribution.

`scale.W`
: `logical`. If `TRUE` (default), the `vcov` method will calculate the pooled covariance matrix by scaling the within-imputation component by the ARIV (see Enders, 2010, p. 235, for definition and formula). Otherwise, the pooled matrix is calculated as the weighted sum of the within-imputation and between-imputation components (see Enders, 2010, ch. 8, for details). This in turn affects how the `summary` method calculates its pooled standard errors, as well as the Wald test (`lavTestWald.mi()`).

`omit.imps`
: `character` vector specifying criteria for omitting imputations from pooled results. Can include any of `c("no.conv", "no.se", "no.npd")`, the first 2 of which are the default setting, which excludes any imputations that did not converge or for which standard errors could not be computed. The last option (`"no.npd"`) would exclude any imputations which yielded a nonpositive definite covariance matrix for observed or latent variables, which would include any "improper solutions" such as Heywood cases. NPD solutions are not excluded by default because they are likely to occur due to sampling error, especially in small samples. However, gross model misspecification could also cause NPD solutions, users can compare pooled results with and without this setting as a sensitivity analysis to see whether some imputations warrant further investigation. Specific imputation numbers can also be included in this argument, in case users want to apply their own custom omission criteria (or simulations can use different numbers of imputations without redundantly refitting the model).

`...`
: Additional arguments passed to `lavTestLRT.mi()`, or subsequently to `lavaan::lavTestLRT()`. This is how users can specify a `pool.method=` for the model's χ^2 statistic (optionally used in any `fit.measures=`), or set `pool.method="D1"` when `summary(modindices=TRUE)`.

`total`
: `logical` (default: `TRUE`) indicating whether the `nobs()` method should return the total sample size or (if `FALSE`) a vector of group sample sizes.

`type`
: The meaning of this argument varies depending on which method it it used for. Find detailed descriptions in the **Value** section under `coef()` and `vcov()`.

`labels`
: `logical` indicating whether the `coef()` output should include parameter labels. Default is `TRUE`.

`baseline.model`, `h1.model`
: See `lavaan::fitMeasures()`.

Value

coef `signature(object = "lavaan.mi", type = "free", labels = TRUE, omit.imps = c("no.conv","no.se"))`: See argument description on the help page for lavaan::lavaan class. Returns the pooled point estimates (i.e., averaged across imputed data sets; see Rubin, 1987).

vcov `signature(object = "lavaan.mi", scale.W = TRUE, omit.imps = c("no.conv","no.se"), type = c("pooled","between","within","ariv"))`: By default, returns the pooled covariance matrix of parameter estimates (`type = "pooled"`), the within-imputations covariance matrix (`type = "within"`), the between-imputations covariance matrix (`type = "between"`), or the average relative increase in variance (`type = "ariv"`) due to missing data.

fitted.values `signature(object = "lavaan.mi", omit.imps = c("no.conv","no.se"))`: See corresponding lavaan::lavaan method. Returns model-implied moments, evaluated at the pooled point estimates.

fitted alias for `fitted.values`

nobs `signature(object = "lavaan.mi", total = TRUE)`: either the total (default) sample size or a vector of group sample sizes (`total = FALSE`).

fitMeasures `signature(object = "lavaan.mi", fit.measures = "all", baseline.model = NULL, h1.model = NULL, fm.args = list(standard.test = "default", scaled.test = "default", rmsea.ci.level = 0.90, rmsea.h0.closefit = 0.05, rmsea.h0.notclosefit = 0.08, robust = TRUE, cat.check.pd = TRUE), output = "vector", omit.imps = c("no.conv","no.se"), ...)`: See `lavaan::fitMeasures()` for details. Pass additional arguments to `lavTestLRT.mi()` via `...`.

fitmeasures alias for `fitMeasures`.

show `signature(object = "lavaan.mi")`: returns a message about convergence rates and estimation problems (if applicable) across imputed data sets.

summary `signature(object = "lavaan.mi", header = TRUE, fit.measures = FALSE,fm.args = list(standard.test = "default", scaled.test = "default", rmsea.ci.level = 0.90, rmsea.h0.closefit = 0.05, rmsea.h0.notclosefit = 0.08, robust = TRUE, cat.check.pd = TRUE), estimates = TRUE, ci = FALSE, standardized = FALSE, std = standardized, cov.std = TRUE, rsquare = FALSE, fmi = FALSE, asymptotic = FALSE, scale.W = !asymptotic, omit.imps = c("no.conv","no.se"), remove.unused = TRUE, modindices = FALSE, nd = 3L, ...)`: Analogous to `summary()` for `lavaan-class` objects. By default, `summary` returns output from `parameterEstimates.mi()`, with some cursory information in the header. Setting `fit.measures=TRUE` will additionally run `fitMeasures()`, and setting `modindices=TRUE` will additionally run `modindices.mi()`.

Slots

`coefList` `list` of estimated coefficients in matrix format (one per imputation) as output by `lavInspect(fit, "est")`

`phiList` `list` of model-implied latent-variable covariance matrices (one per imputation) as output by `lavInspect(fit, "cov.lv")`

`miList` `list` of modification indices output by `lavaan::modindices()`

`lavListCall` call to `lavaan::lavaanList()` used to fit the model to the list of imputed data sets in `@DataList`, stored as a `list` of arguments

`convergence` `list` of `logical` vectors indicating whether, for each imputed data set, (1) the model converged on a solution, (2) *SE*s could be calculated, (3) the (residual) covariance matrix of latent variables (Ψ) is non-positive-definite, and (4) the residual covariance matrix of observed variables (Θ) is non-positive-definite.

`version` Named `character` vector indicating the `lavaan` and `lavaan.mi` version numbers.

`DataList` The `list` of imputed data sets

`SampleStatsList` List of output from `lavInspect(fit, "sampstat")` applied to each fitted model.

`ParTableList,vcovList,testList,baselineList` See lavaan::lavaanList

`h1List` See lavaan::lavaanList. An additional element is added to the `list`: `$PT` is the "saturated" model's parameter table, returned by `lavaan::lav_partable_unrestricted()`.

`call,Options,ParTable,pta,Data,Model,meta,timingList,CacheList,optimList,impliedList,loglikList,int`
By default, `lavaan.mi()` does not populate the remaining `@*List` slots from the lavaan::lavaanList class. But they can be added to the call using the `store.slots=` argument (passed to `lavaan::lavaanList()` via ...).

Objects from the Class

See the `lavaan.mi()` function for details. Wrapper functions include `cfa.mi()`, `sem.mi()`, and `growth.mi()`.

Author(s)

Terrence D. Jorgensen (University of Amsterdam; <TJorgensen314@gmail.com>)

References

Enders, C. K. (2010). *Applied missing data analysis*. New York, NY: Guilford.

Rubin, D. B. (1987). *Multiple imputation for nonresponse in surveys*. New York, NY: Wiley. doi:10.1002/9780470316696

Examples

```
data(HS20imps) # import a list of 20 imputed data sets

## specify CFA model from lavaan's ?cfa help page
HS.model <- '
  visual  =~ x1 + x2 + x3
  textual =~ x4 + x5 + x6
  speed   =~ x7 + x8 + x9
'

## fit model to imputed data sets
fit <- cfa.mi(HS.model, data = HS20imps)

## vector of pooled coefficients
coef(fit)
```

```
## their pooled asymptotic covariance matrix
vcov(fit)
## which is the weighted sum of within- and between-imputation components
vcov(fit, type = "within")
vcov(fit, type = "between")

## covariance matrix of observed variables,
## as implied by pooled estimates
fitted(fit)

## custom null model for CFI
HS.parallel <- '
  visual  =~ x1 + 1*x2 + 1*x3
  textual =~ x4 + 1*x5 + 1*x6
  speed   =~ x7 + 1*x8 + 1*x9
'
fit0 <- cfa.mi(HS.parallel, data = HS20imps, orthogonal = TRUE)
fitMeasures(fit, baseline.model = fit0, fit.measures = "default",
            output = "text")

## See ?lavaan.mi help page for more examples
```

`lavResiduals.mi` *Covariance and Correlation Residuals*

Description

This function calculates residuals for sample moments (e.g., means and (co)variances, means) from a lavaan model fitted to multiple imputed data sets, along with summary and inferential statistics about the residuals.

Usage

```
## S4 method for signature 'lavaan.mi'
residuals(object, type = "raw", omit.imps = c("no.conv", "no.se"), ...)

## S4 method for signature 'lavaan.mi'
resid(object, type = "raw", omit.imps = c("no.conv", "no.se"), ...)

lavResiduals.mi(object, omit.imps = c("no.conv", "no.se"), ...)
```

Arguments

`object`	An object of class `lavaan.mi`
`type`	`character` indicating whether/how to standardize the covariance residuals. If `type = "raw"`, the raw (= unscaled) difference between the observed and expected (model-implied) summary statistics are returned. The observed summary statistics are averaged across imputations, and the model-implied statistics are

	calculated from pooled parameter estimates (as returned by `fitted.values()`). If `type = "cor"` or `"cor.bollen"`, the observed and model-implied covariance matrices are first transformed to correlation matrices (using `stats::cov2cor()`); then correlation residuals are computed. If `type = "cor.bentler"`, both the observed and model-implied covariance matrices are rescaled by dividing the elements by the square roots of the corresponding variances of the observed covariance matrix.
`omit.imps`	`character` indicating criteria for excluding imputations from pooled results. See lavaan.mi for argument details.
`...`	Arguments passed to `lavaan::lavResiduals()`.

Value

A `list` of residuals and other information (see `lavaan::lavResiduals()`). The standard `residuals()` (and `resid()` alias) method simply calls `lavResiduals.mi(..., zstat=FALSE, summary=FALSE)`.

Author(s)

Terrence D. Jorgensen (University of Amsterdam; <TJorgensen314@gmail.com>)

See Also

`lavaan::lavResiduals()` for details about other arguments.

Examples

```
data(HS20imps) # import a list of 20 imputed data sets

## specify CFA model from lavaan's ?cfa help page
HS.model <- '
  visual  =~ x1 + x2 + x3
  textual =~ x4 + x5 + x6
  speed   =~ x7 + x8 + x9
'
## fit model to 20 imputed data sets
fit <- cfa.mi(HS.model, data = HS20imps)

## default type = "cor.bentler" (standardized covariance residuals)
lavResiduals.mi(fit, zstat = FALSE)
## SRMR is in the $summary

## correlation residuals
lavResiduals.mi(fit, zstat = FALSE, type = "cor")
## CRMR is in the $summary

## raw covariance residuals
lavResiduals.mi(fit, type = "raw") # zstat=TRUE by default
## RMR is in the $summary
## "normalized" residuals are in $cov.z
```

```
## The standard resid() and residuals() method simply call lavResiduals.mi()
## with arguments to display only the residuals ("raw" by default).

resid(fit)
residuals(fit, type = "cor.bollen") # same as type = "cor"
```

`lavTestLRT.mi` *Likelihood Ratio Test for Multiple Imputations*

Description

Likelihood ratio test (LRT) for lavaan models fitted to multiple imputed data sets.

Usage

```
lavTestLRT.mi(
  object,
  ...,
  modnames = NULL,
  asANOVA = TRUE,
  pool.method = c("D4", "D3", "D2"),
  omit.imps = c("no.conv", "no.se"),
  asymptotic = FALSE,
  pool.robust = FALSE
)

## S4 method for signature 'lavaan.mi'
anova(object, ...)
```

Arguments

`object`	An object of class lavaan.mi
`...`	Additional objects of class lavaan.mi, as well as arguments passed to `lavaan::lavTestLRT()` when `pool.method = "D2"` and `pool.robust = TRUE`.
`modnames`	Optional `character` of model names to use as row names in the resulting matrix of results (when more than 2 models are compared)
`asANOVA`	`logical` indicating whether to return an object of class `"anova"`. If `FALSE`, a numeric vector is returned for one (pair of) model(s), or a `data.frame` is returned for multiple pairs of models.
`pool.method`	`character` indicating which pooling method to use. • `"D4"`, `"new.LRT"`, `"cm"`, or `"chan.meng"` requests the method described by Chan & Meng (2022). This is currently the default. • `"D3"`, `"old.LRT"`, `"mr"`, or `"meng.rubin"` requests the method described by Meng & Rubin (1992).

- "D2", "LMRR", or "Li.et.al" requests the complete-data LRT statistic should be calculated using each imputed data set, which will then be pooled across imputations, as described in Li, Meng, Raghunathan, & Rubin (1991).

Find additional details in Enders (2010, chapter 8).

omit.imps — character vector specifying criteria for omitting imputations from pooled results. Can include any of c("no.conv", "no.se", "no.npd"), the first 2 of which are the default setting, which excludes any imputations that did not converge or for which standard errors could not be computed. The last option ("no.npd") would exclude any imputations which yielded a nonpositive definite covariance matrix for observed or latent variables, which would include any "improper solutions" such as Heywood cases. Specific imputation numbers can also be included in this argument, in case users want to apply their own custom omission criteria (or simulations can use different numbers of imputations without redundantly refitting the model).

asymptotic — logical. If FALSE (default), the pooled test will be returned as an F-distributed statistic with numerator (df1) and denominator (df2) degrees of freedom. If TRUE, the pooled F statistic will be multiplied by its df1 on the assumption that its df2 is sufficiently large enough that the statistic will be asymptotically χ^2 distributed with df1.

pool.robust — logical. Ignored unless pool.method = "D2" and a robust test was requested. If pool.robust = TRUE, the robust test statistic is pooled, whereas pool.robust = FALSE will pool the naive test statistic (or difference statistic) and apply the average scale/shift parameter to it. The harmonic mean is applied to the scaling factor, whereas the arithmetic mean is applied to the shift parameter.

Details

The "D2" method is available using any estimator and test statistic. When using a likelihood-based estimator, 2 additional methods are available to pool the LRT.

- The Meng & Rubin (1992) method, commonly referred to as "D3". This method has many problems, discussed in Chan & Meng (2022).
- The Chan & Meng (2022) method, referred to as "D4" by Grund et al. (2023), resolves problems with "D3".

When "D2" is not explicitly requested in situations it is the only applicable method, (e.g., DWLS for categorical outcomes), users are notified that pool.method was set to "D2".

pool.method = "Mplus" implies "D3" and asymptotic = TRUE (see Asparouhov & Muthen, 2010).

Note that the anova() method simply calls lavTestLRT.mi().

Value

- When asANOVA=TRUE, returns an object of class stats::anova with a a test of model fit for a single model (object) or test(s) of the difference(s) in fit between nested models passed via ... (either an F or χ^2 statistic, depending on the asymptotic argument), its degrees of freedom, its p value, and 2 missing-data diagnostics: the relative increase in variance (RIV = FMI / (1 − FMI)) and the fraction of missing information (FMI = RIV / (1 + RIV)).

- When `asANOVA=FALSE`, returns a vector containing the LRT statistic for a single model or comparison of a single pair of models, or a `data.frame` of multiple model comparisons. Robust statistics will also include the average (across imputations) scaling factor and (if relevant) shift parameter(s), unless `pool.robust = TRUE`. When using `pool.method = "D3"` or `"D4"`, the vector for a single model also includes its average log-likelihood and information criteria.

Author(s)

Terrence D. Jorgensen (University of Amsterdam; <TJorgensen314@gmail.com>)

Based on source code for `lavaan::lavTestLRT()` by Yves Rosseel.

References

Asparouhov, T., & Muthen, B. (2010). *Chi-square statistics with multiple imputation*. Technical Report. Retrieved from http://www.statmodel.com/

Chan, K. W., & Meng, X. L. (2022). Multiple improvements of multiple imputation likelihood ratio tests. *Statistica Sinica, 32*, 1489–1514. doi:10.5705/ss.202019.0314

Enders, C. K. (2010). *Applied missing data analysis*. New York, NY: Guilford.

Grund, S., Lüdtke, O., & Robitzsch, A. (2023). Pooling methods for likelihood ratio tests in multiply imputed data sets. *Psychological Methods, 28*(5), 1207–1221. doi:10.1037/met0000556

Li, K.-H., Meng, X.-L., Raghunathan, T. E., & Rubin, D. B. (1991). Significance levels from repeated *p*-values with multiply-imputed data. *Statistica Sinica, 1*(1), 65–92. Retrieved from https://www.jstor.org/stable/24303994

Meng, X.-L., & Rubin, D. B. (1992). Performing likelihood ratio tests with multiply-imputed data sets. *Biometrika, 79*(1), 103–111. doi:10.2307/2337151

Rubin, D. B. (1987). *Multiple imputation for nonresponse in surveys*. New York, NY: Wiley. doi:10.1002/9780470316696

See Also

`lavaan::lavTestLRT()` for arguments that can be passed via . . . , and use `lavaan::fitMeasures()` to obtain fit indices calculated from pooled test statistics.

Examples

```
data(HS20imps) # import a list of 20 imputed data sets

## specify CFA model from ?lavaan::cfa help page
HS.model <- '
  visual  =~ x1 + x2 + x3
  textual =~ x4 + x5 + x6
  speed   =~ x7 + x8 + x9
'

fit1 <- cfa.mi(HS.model, data = HS20imps, estimator = "mlm")

## By default, pool.method = "D4".
## Must request an asymptotic chi-squared statistic
```

```
## in order to accommodate a robust correction.
lavTestLRT.mi(fit1, asymptotic = TRUE)
## or   anova(fit1, asymptotic = TRUE)

## Comparison with more constrained (nested) models: parallel indicators
HS.parallel <- '
  visual  =~ x1 + 1*x2 + 1*x3
  textual =~ x4 + 1*x5 + 1*x6
  speed   =~ x7 + 1*x8 + 1*x9
'

fitp <- cfa.mi(HS.parallel, data = HS20imps, estimator = "mlm")

## Even more constrained model: orthogonal factors
fit0 <- cfa.mi(HS.parallel, data = HS20imps, estimator = "mlm",
               orthogonal = TRUE)

## Compare 3 models, and pass the lavTestLRT(method=) argument

lavTestLRT.mi(fit1, fit0, fitp, asymptotic = TRUE,
              method = "satorra.bentler.2010")

## For a single model, you can request a vector instead of an anova-class
## table in order to see naive information criteria (only using D3 or D4),
## which are calculated using the average log-likelihood across imputations.
lavTestLRT.mi(fit1, asANOVA = FALSE)



## When using a least-squares (rather than maximum-likelihood) estimator,
## only the D2 method is available.  For example, ordered-categorical data:
data(binHS5imps) # import a list of 5 imputed data sets

## fit model using default DWLS estimation
fit1c <- cfa.mi(HS.model   , data = binHS5imps, ordered = TRUE)
fit0c <- cfa.mi(HS.parallel, data = binHS5imps, ordered = TRUE,
                orthogonal = TRUE)

## Using D2, you can either robustify the pooled naive statistic ...
lavTestLRT.mi(fit1c, fit0c, asymptotic = TRUE, pool.method = "D2")
## ... or pool the robust chi-squared statistic (NOT recommended)
lavTestLRT.mi(fit1c, fit0c, asymptotic = TRUE, pool.method = "D2",
              pool.robust = TRUE)

## When calculating fit indices, you can pass lavTestLRT.mi() arguments:
fitMeasures(fit1c, output = "text",
            # lavTestLRT.mi() arguments:
            pool.method = "D2", pool.robust = TRUE)
```

lavTestScore.mi *Score Test for Multiple Imputations*

Description

Score test (or "Lagrange multiplier" test) for lavaan models fitted to multiple imputed data sets. Statistics for releasing one or more fixed or constrained parameters in model can be calculated by pooling the gradient and information matrices pooled across imputed data sets in a method proposed by Mansolf, Jorgensen, & Enders (2020)—analogous to the "D1" Wald test proposed by Li, Meng, Raghunathan, & Rubin's (1991)—or by pooling the complete-data score-test statistics across imputed data sets (i.e., "D2"; Li et al., 1991).

Usage

```
lavTestScore.mi(
  object,
  add = NULL,
  release = NULL,
  pool.method = c("D2", "D1"),
  scale.W = !asymptotic,
  omit.imps = c("no.conv", "no.se"),
  asymptotic = is.null(add),
  univariate = TRUE,
  cumulative = FALSE,
  epc = FALSE,
  standardized = epc,
  cov.std = epc,
  verbose = FALSE,
  warn = TRUE,
  information = "expected"
)
```

Arguments

object	An object of class lavaan.mi.
add	Either a character string (typically between single quotes) or a parameter table containing additional (currently fixed-to-zero) parameters for which the score test must be computed.
release	Vector of integers. The indices of the *equality* constraints that should be released. The indices correspond to the order of the equality constraints as they appear in the parameter table.
pool.method	character indicating which pooling method to use. "D1" requests Mansolf, Jorgensen, & Enders' (2020) proposed Wald-like test for pooling the gradient and information, which are then used to calculate score-test statistics in the usual manner. "D2" (default because it is less computationall intensive) requests to pool the complete-data score-test statistics from each imputed data set, then pool them across imputations, described by Li et al. (1991) and Enders (2010).

`scale.W` `logical`. If `FALSE`, the pooled information matrix is calculated as the weighted sum of the within-imputation and between-imputation components. Otherwise, the pooled information is calculated by scaling the within-imputation component by the average relative increase in variance (ARIV; Enders, 2010, p. 235), which is *only* consistent when requesting the F test (i.e., `asymptotic = FALSE`. Ignored (irrelevant) if `pool.method = "D2"`.

`omit.imps` `character` vector specifying criteria for omitting imputations from pooled results. Can include any of `c("no.conv", "no.se", "no.npd")`, the first 2 of which are the default setting, which excludes any imputations that did not converge or for which standard errors could not be computed. The last option (`"no.npd"`) would exclude any imputations which yielded a nonpositive definite covariance matrix for observed or latent variables, which would include any "improper solutions" such as Heywood cases. Specific imputation numbers can also be included in this argument, in case users want to apply their own custom omission criteria (or simulations can use different numbers of imputations without redundantly refitting the model).

`asymptotic` `logical`. If `FALSE` (default when using `add` to test adding fixed parameters to the model), the pooled test will be returned as an F-distributed variable with numerator (`df1`) and denominator (`df2`) degrees of freedom. If `TRUE`, the pooled F statistic will be multiplied by its `df1` on the assumption that its `df2` is sufficiently large enough that the statistic will be asymptotically χ^2 distributed with `df1`. When using the `release` argument, `asymptotic` will be set to `TRUE` because (A)RIV can only be calculated for added parameters.

`univariate` `logical`. If `TRUE`, compute the univariate score statistics, one for each constraint.

`cumulative` `logical`. If `TRUE`, order the univariate score statistics from large to small, and compute a series of multivariate score statistics, each time including an additional constraint in the test.

`epc` `logical`. If `TRUE`, and we are releasing existing constraints, compute the expected parameter changes for the existing (free) parameters (and any specified with `add`), if all constraints were released. For EPCs associated with a particular (1-*df*) constraint, only specify one parameter in `add` or one constraint in `release`.

`standardized` If `TRUE`, two extra columns (`sepc.lv` and `sepc.all`) in the `$epc` table will contain standardized values for the EPCs. See `lavaan::lavTestScore()`.

`cov.std` `logical`. See `lavaan::standardizedSolution()`.

`verbose` `logical`. Not used for now.

`warn` `logical`. If `TRUE`, print warnings if they occur.

`information` `character` indicating the type of information matrix to use (check `lavaan::lavInspect()` for available options). `"expected"` information is the default, which provides better control of Type I errors.

Value

A list containing at least one `data.frame`:

- `$test`: The total score test, with columns for the score test statistic (`X2`), its degrees of freedom (`df`), its *p* value under the χ^2 distribution (`p.value`), and if `asymptotic=FALSE`, the average relative invrease in variance (ARIV) used to calculate the denominator *df* is also returned as a missing-data diagnostic, along with the fraction missing information (FMI = ARIV / (1 + ARIV)).
- `$uni`: Optional (if `univariate=TRUE`). Each 1-*df* score test, equivalent to modification indices. Also includes EPCs if `epc=TRUE`, and RIV and FMI if `asymptotic=FALSE`.
- `$cumulative`: Optional (if `cumulative=TRUE`). Cumulative score tests, with ARIV and FMI if `asymptotic=FALSE`.
- `$epc`: Optional (if `epc=TRUE`). Parameter estimates, expected parameter changes, and expected parameter values if ALL the tested constraints were freed.

See `lavaan::lavTestScore()` for details.

Author(s)

Terrence D. Jorgensen (University of Amsterdam; <TJorgensen314@gmail.com>)

Based on source code for `lavaan::lavTestScore()` by Yves Rosseel

`pool.method = "D1"` method proposed by Maxwell Mansolf (University of California, Los Angeles; <mamansolf@gmail.com>)

References

Bentler, P. M., & Chou, C.-P. (1992). Some new covariance structure model improvement statistics. *Sociological Methods & Research, 21*(2), 259–282. doi:10.1177/0049124192021002006

Enders, C. K. (2010). *Applied missing data analysis*. New York, NY: Guilford.

Li, K.-H., Meng, X.-L., Raghunathan, T. E., & Rubin, D. B. (1991). Significance levels from repeated *p*-values with multiply-imputed data. *Statistica Sinica, 1*(1), 65–92. Retrieved from https://www.jstor.org/stable/24303994

Mansolf, M., Jorgensen, T. D., & Enders, C. K. (2020). A multiple imputation score test for model modification in structural equation models. *Psychological Methods, 25*(4), 393–411. doi:10.1037/met0000243

See Also

`lavaan::lavTestScore()`

Examples

```
data(HS20imps) # import a list of 20 imputed data sets

## specify CFA model from lavaan's ?cfa help page
HS.model <- '
  speed =~ c(L1, L1)*x7 + c(L1, L1)*x8 + c(L1, L1)*x9
'

fit <- cfa.mi(HS.model, data = HS20imps, group = "school", std.lv = TRUE)
```

```
## Mode 1: Score test for releasing equality constraints

## default test: Li et al.'s (1991) "D2" method
lavTestScore.mi(fit, cumulative = TRUE)

## Li et al.'s (1991) "D1" method,
## adapted for score tests by Mansolf et al. (2020)

lavTestScore.mi(fit, pool.method = "D1")


## Mode 2: Score test for adding currently fixed-to-zero parameters
lavTestScore.mi(fit, add = 'x7 ~~ x8 + x9')
```

`lavTestWald.mi` *Wald Test for Multiple Imputations*

Description

Wald test for testing a linear hypothesis about the parameters of lavaan models fitted to multiple imputed data sets. Statistics for constraining one or more free parameters in a model can be calculated from the pooled point estimates and asymptotic covariance matrix of model parameters using Rubin's (1987) rules, or by pooling the Wald test statistics across imputed data sets (Li, Meng, Raghunathan, & Rubin, 1991).

Usage

```
lavTestWald.mi(
  object,
  constraints = NULL,
  pool.method = c("D1", "D2"),
  asymptotic = FALSE,
  scale.W = !asymptotic,
  omit.imps = c("no.conv", "no.se"),
  verbose = FALSE,
  warn = TRUE
)
```

Arguments

`object`	An object of class lavaan.mi.
`constraints`	A `character` string (typically between single quotes) containing one or more equality constraints. See examples for more details
`pool.method`	`character` indicating which pooling method to use. `"D1"` or `"Rubin"` (default) indicates Rubin's (1987) rules will be applied to the point estimates and the asymptotic covariance matrix of model parameters, and those pooled values

will be used to calculate the Wald test in the usual manner. "D2", "LMRR", or "Li.et.al" indicate that the complete-data Wald test statistic should be calculated using each imputed data set, which will then be pooled across imputations, as described in Li, Meng, Raghunathan, & Rubin (1991) and Enders (2010, chapter 8).

asymptotic — logical. If FALSE (default), the pooled test will be returned as an F-distributed statistic with numerator (df1) and denominator (df2) degrees of freedom. If TRUE, the pooled F statistic will be multiplied by its df1 on the assumption that its df2 is sufficiently large enough that the statistic will be asymptotically χ^2 distributed with df1.

scale.W — logical. If FALSE, the pooled asymptotic covariance matrix of model parameters is calculated as the weighted sum of the within-imputation and between-imputation components. Otherwise, the pooled asymptotic covariance matrix of model parameters is calculated by scaling the within-imputation component by the average relative increase in variance (ARIV; see Enders, 2010, p. 235), which is *only* consistent when requesting the F test (i.e., asymptotic = FALSE. Ignored (irrelevant) if pool.method = "D2".

omit.imps — character vector specifying criteria for omitting imputations from pooled results. Can include any of c("no.conv", "no.se", "no.npd"), the first 2 of which are the default setting, which excludes any imputations that did not converge or for which standard errors could not be computed. The last option ("no.npd") would exclude any imputations which yielded a nonpositive definite covariance matrix for observed or latent variables, which would include any "improper solutions" such as Heywood cases. Specific imputation numbers can also be included in this argument, in case users want to apply their own custom omission criteria (or simulations can use different numbers of imputations without redundantly refitting the model).

verbose — logical. If TRUE, print the restriction matrix and the estimated restricted values.

warn — logical. If TRUE, print warnings if they occur.

Details

The constraints are specified using the "==" operator. Both the left-hand side and the right-hand side of the equality can contain a linear combination of model parameters, or a constant (like zero). The model parameters must be specified by their user-specified labels from the link[lavaan]{model.syntax}. Names of defined parameters (using the ":=" operator) can be included too.

Value

A vector containing the Wald test statistic (either an F or χ^2 statistic, depending on the asymptotic argument), the degrees of freedom (numerator and denominator, if asymptotic = FALSE), and a p value. If asymptotic = FALSE, the relative invrease in variance (RIV, or average for multiparameter tests: ARIV) used to calculate the denominator *df* is also returned as a missing-data diagnostic, along with the fraction missing information (FMI = ARIV / (1 + ARIV)).

Author(s)

Terrence D. Jorgensen (University of Amsterdam; <TJorgensen314@gmail.com>)

Based on source code for `lavaan::lavTestWald()` by Yves Rosseel

References

Enders, C. K. (2010). *Applied missing data analysis*. New York, NY: Guilford.

Li, K.-H., Meng, X.-L., Raghunathan, T. E., & Rubin, D. B. (1991). Significance levels from repeated *p*-values with multiply-imputed data. *Statistica Sinica, 1*(1), 65–92. Retrieved from https://www.jstor.org/stable/24303994

Rubin, D. B. (1987). *Multiple imputation for nonresponse in surveys*. New York, NY: Wiley. doi:10.1002/9780470316696

See Also

`lavaan::lavTestWald()`

Examples

```
data(HS20imps) # import a list of 20 imputed data sets

## specify CFA model from lavaan's ?cfa help page
HS.model <- '
  visual  =~ x1 + b1*x2 + x3
  textual =~ x4 + b2*x5 + x6
  speed   =~ x7 + b3*x8 + x9
'

fit <- cfa.mi(HS.model, data = HS20imps)

## Testing whether a single parameter equals zero yields the 'chi-square'
## version of the Wald z statistic from the summary() output, or the
## 'F' version of the t statistic from the summary() output, depending
## whether asymptotic = TRUE or FALSE
lavTestWald.mi(fit, constraints = "b1 == 0")             # default D1 statistic
lavTestWald.mi(fit, constraints = "b1 == 0", pool.method = "D2") # D2 statistic

## The real advantage is simultaneously testing several equality
## constraints, or testing more complex constraints:
con <- '
   2*b1 == b3
   b2 - b3 == 0
'
lavTestWald.mi(fit, constraints = con) # default F statistic
lavTestWald.mi(fit, constraints = con, asymptotic = TRUE) # chi-squared
```

`modindices.mi` *Modification Indices for Multiple Imputations*

Description

Modification indices (1-*df* Lagrange multiplier tests) from a latent variable model fitted to multiple imputed data sets. Statistics for releasing one or more fixed or constrained parameters in model can be calculated by pooling the gradient and information matrices across imputed data sets in a method proposed by Mansolf, Jorgensen, & Enders (2020)—analogous to the "D1" Wald test proposed by Li, Meng, Raghunathan, & Rubin (1991)—or by pooling the complete-data score-test statistics across imputed data sets (i.e., "D2"; Li et al., 1991).

Usage

```
modindices.mi(
  object,
  pool.method = c("D2", "D1"),
  omit.imps = c("no.conv", "no.se"),
  standardized = TRUE,
  cov.std = TRUE,
  information = "expected",
  power = FALSE,
  delta = 0.1,
  alpha = 0.05,
  high.power = 0.75,
  sort. = FALSE,
  minimum.value = 0,
  maximum.number = nrow(LIST),
  na.remove = TRUE,
  op = NULL
)

modificationIndices.mi(
  object,
  pool.method = c("D2", "D1"),
  omit.imps = c("no.conv", "no.se"),
  standardized = TRUE,
  cov.std = TRUE,
  information = "expected",
  power = FALSE,
  delta = 0.1,
  alpha = 0.05,
  high.power = 0.75,
  sort. = FALSE,
  minimum.value = 0,
  maximum.number = nrow(LIST),
  na.remove = TRUE,
```

```
  op = NULL
)

modificationindices.mi(
  object,
  pool.method = c("D2", "D1"),
  omit.imps = c("no.conv", "no.se"),
  standardized = TRUE,
  cov.std = TRUE,
  information = "expected",
  power = FALSE,
  delta = 0.1,
  alpha = 0.05,
  high.power = 0.75,
  sort. = FALSE,
  minimum.value = 0,
  maximum.number = nrow(LIST),
  na.remove = TRUE,
  op = NULL
)
```

Arguments

`object`	An object of class lavaan.mi
`pool.method`	`character` indicating which pooling method to use. `"D1"` requests Mansolf, Jorgensen, & Enders' (2020) proposed Wald-like test for pooling the gradient and information, which are then used to calculate score-test statistics in the usual manner. `"D2"` (default because it is less computationally intensive) requests to pool the complete-data score-test statistics from each imputed data set, then pool them across imputations, described by Li et al. (1991) and Enders (2010).
`omit.imps`	`character` vector specifying criteria for omitting imputations from pooled results. Can include any of `c("no.conv", "no.se", "no.npd")`, the first 2 of which are the default setting, which excludes any imputations that did not converge or for which standard errors could not be computed. The last option (`"no.npd"`) would exclude any imputations which yielded a nonpositive definite covariance matrix for observed or latent variables, which would include any "improper solutions" such as Heywood cases. Specific imputation numbers can also be included in this argument, in case users want to apply their own custom omission criteria (or simulations can use different numbers of imputations without redundantly refitting the model).
`standardized`	`logical`. If `TRUE`, two extra columns (`$sepc.lv` and `$sepc.all`) will contain standardized values for the EPCs. In the first column (`$sepc.lv`), standardization is based on the variances of the (continuous) latent variables. In the second column (`$sepc.all`), standardization is based on both the variances of both (continuous) observed and latent variables. (Residual) covariances are standardized using (residual) variances.
`cov.std`	`logical`. `TRUE` if `pool.method == "D2"`. If `TRUE` (default), the (residual) observed covariances are scaled by the square-root of the diagonal elements of

	the Θ matrix, and the (residual) latent covariances are scaled by the square-root of the diagonal elements of the Ψ matrix. If `FALSE`, the (residual) observed covariances are scaled by the square-root of the diagonal elements of the model-implied covariance matrix of observed variables (Σ), and the (residual) latent covariances are scaled by the square-root of the diagonal elements of the model-implied covariance matrix of the latent variables.
`information`	`character` indicating the type of information matrix to use (check `lavaan::lavInspect()` for available options). `"expected"` information is the default, which provides better control of Type I errors.
`power`	`logical`. If `TRUE`, the (post-hoc) power is computed for each modification index, using the values of `delta` and `alpha`.
`delta`	The value of the effect size, as used in the post-hoc power computation, currently using the unstandardized metric of the `$epc` column.
`alpha`	The significance level used for deciding if the modification index is statistically significant or not.
`high.power`	If the computed power is higher than this cutoff value, the power is considered 'high'. If not, the power is considered 'low'. This affects the values in the `$decision` column in the output.
`sort.`	`logical`. If `TRUE`, sort the output using the values of the modification index values. Higher values appear first.
`minimum.value`	`numeric`. Filter output and only show rows with a modification index value equal or higher than this minimum value.
`maximum.number`	`integer`. Filter output and only show the first maximum number rows. Most useful when combined with the `sort.` option.
`na.remove`	`logical`. If `TRUE` (default), filter output by removing all rows with `NA` values for the modification indices.
`op`	`character` string. Filter the output by selecting only those rows with operator `op`.

Value

A `data.frame` containing modification indices and (S)EPCs.

Note

When `pool.method = "D2"`, each (S)EPC will be pooled by taking its average across imputations. When `pool.method = "D1"`, EPCs will be calculated in the standard way using the pooled gradient and information, and SEPCs will be calculated by standardizing the EPCs using model-implied (residual) variances.

Author(s)

Terrence D. Jorgensen (University of Amsterdam; <TJorgensen314@gmail.com>)

Based on source code for `lavaan::modindices()` by Yves Rosseel

`pool.method = "D1"` method proposed by Maxwell Mansolf (University of California, Los Angeles; <mamansolf@gmail.com>)

References

Enders, C. K. (2010). *Applied missing data analysis*. New York, NY: Guilford.

Li, K.-H., Meng, X.-L., Raghunathan, T. E., & Rubin, D. B. (1991). Significance levels from repeated *p*-values with multiply-imputed data.*Statistica Sinica, 1*(1), 65–92. Retrieved from https://www.jstor.org/stable/24303994

Mansolf, M., Jorgensen, T. D., & Enders, C. K. (2020). A multiple imputation score test for model modification in structural equation models. *Psychological Methods, 25*(4), 393–411. doi:10.1037/met0000243

See Also

lavTestScore.mi()

Examples

```
data(HS20imps) # import a list of 20 imputed data sets

## specify CFA model from lavaan's ?cfa help page
HS.model <- '
  visual  =~ x1 + x2 + x3
  textual =~ x4 + x5 + x6
  speed   =~ x7 + x8 + x9
'

fit <- cfa.mi(HS.model, data = HS20imps)

modindices.mi(fit) # default: Li et al.'s (1991) "D2" method

## Li et al.'s (1991) "D1" method,
## adapted for score tests by Mansolf et al. (2020)
modindices.mi(fit, pool.method = "D1")
```

---

parameterEstimates.mi *Pooled Parameter Estimates*

Description

This function pools parameter estimates from a lavaan model fitted to multiple imputed data sets.

Usage

```
parameterEstimates.mi(
  object,
  se = TRUE,
  zstat = se,
  pvalue = zstat,
```

```
  ci = TRUE,
  level = 0.95,
  fmi = FALSE,
  standardized = FALSE,
  cov.std = TRUE,
  rsquare = FALSE,
  asymptotic = FALSE,
  scale.W = !asymptotic,
  omit.imps = c("no.conv", "no.se"),
  remove.system.eq = TRUE,
  remove.eq = TRUE,
  remove.ineq = TRUE,
  remove.def = FALSE,
  remove.nonfree = FALSE,
  remove.unused = FALSE,
  output = "data.frame",
  header = FALSE
)
```

Arguments

`object`	An object of class `lavaan.mi`
`se`, `zstat`, `pvalue`, `ci`, `level`, `standardized`, `cov.std`, `rsquare`, `remove.system.eq`, `remove.eq`, `remove.ineq`, `remove.def`, `remove.nonfree`, `remove.unused`, `output`, `header`	See `lavaan::parameterEstimates()`.
`fmi`	`logical` indicating whether to add 2 columns: • the fraction of missing information (`$fmi`), which is the ratio of between-imputation variance to total (pooled) sampling variance • the relative increase in variance (`$riv`), which is the ratio of between-imputation variance to within-imputation variance Thus, RIV = FMI / (1 − FMI) and FMI = RIV / (1 + RIV). Ignored when `se=FALSE`.
`asymptotic`	`logical`. When `FALSE`, pooled Wald tests will be *t* statistics with associated degrees of freedom (*df*). When `TRUE`, the *df* are assumed to be sufficiently large for a *t* statistic to approximate a standard normal distribution, so it is printed as a *z* statistic.
`scale.W`	`logical`. If `TRUE` (default), the `vcov` method will calculate the pooled covariance matrix by scaling the within-imputation component by the ARIV (see Enders, 2010, p. 235, for definition and formula). Otherwise, the pooled matrix is calculated as the weighted sum of the within-imputation and between-imputation components (see Enders, 2010, ch. 8, for details).
`omit.imps`	`character` indicating criteria for excluding imputations from pooled results. See lavaan.mi for argument details.

Value

A `data.frame`, analogous to `lavaan::parameterEstimates()`, but estimates, *SE*s, and tests are pooled across imputations.

Author(s)

Terrence D. Jorgensen (University of Amsterdam; <TJorgensen314@gmail.com>)

References

Enders, C. K. (2010). *Applied missing data analysis*. New York, NY: Guilford.

Rubin, D. B. (1987). *Multiple imputation for nonresponse in surveys*. New York, NY: Wiley. doi:10.1002/9780470316696

See Also

`standardizedSolution.mi()` to obtain inferential statistics for pooled standardized parameter estimates.

Examples

```
data(HS20imps) # import a list of 20 imputed data sets

## specify CFA model from lavaan's ?cfa help page
HS.model <- '
  visual  =~ x1 + x2 + x3
  textual =~ x4 + x5 + x6
  speed   =~ x7 + x8 + x9
'
## fit model to 20 imputed data sets
fit <- cfa.mi(HS.model, data = HS20imps)

## pooled estimates, with various optional features:

parameterEstimates.mi(fit, asymptotic = TRUE, rsquare = TRUE)
parameterEstimates.mi(fit, ci = FALSE, fmi = TRUE, output = "text")
parameterEstimates.mi(fit, standardized = "std.all", se = FALSE)
```

---

poolSat *Fit a Saturated* lavaan *Model to Multiple Imputed Data Sets*

Description

This function fits a saturated model to a list of imputed data sets, and returns a list of pooled summary statistics to treat as data.

Usage

```
poolSat(
  data,
  ...,
  return.fit = FALSE,
  scale.W = TRUE,
  omit.imps = c("no.conv", "no.se")
)
```

Arguments

data	A list of imputed data sets, or an object class from which imputed data can be extracted. Recognized classes are lavaan.mi (list of imputations stored in the @DataList slot), amelia (created by the Amelia package), or mids (created by the mice package).
...	Additional arguments passed to lavaan::lavCor() or to lavaan.mi().
return.fit	logical indicating whether to return a lavaan.mi object containing the results of fitting the saturated model to multiple imputed data. Could be useful for diagnostic purposes.
scale.W	logical. If TRUE (default), the within- and between-imputation components will be pooled by scaling the within-imputation component by the ARIV (see Enders, 2010, p. 235, for definition and formula). Otherwise, the pooled matrix is calculated as the weighted sum of the within-imputation and between-imputation components (see Enders, 2010, ch. 8, for details).
omit.imps	character vector specifying criteria for omitting imputations from pooled results of saturated model. Can include any of c("no.conv", "no.se", "no.npd"), the first 2 of which are the default setting, which excludes any imputations that did not converge or for which standard errors could not be computed. The last option ("no.npd") would exclude any imputations which yielded a nonpositive definite covariance matrix for observed or latent variables, which would include any "improper solutions" such as Heywood cases. NPD solutions are not excluded by default because they are likely to occur due to sampling error, especially in small samples. However, gross model misspecification could also cause NPD solutions, users can compare pooled results with and without this setting as a sensitivity analysis to see whether some imputations warrant further investigation. Specific imputation numbers can also be included in this argument, in case users want to apply their own custom omission criteria (or simulation studies can use different numbers of imputations without redundantly refitting the model).

Value

If return.fit=TRUE, a lavaan.mi object. Otherwise, an object of class lavMoments, which is a list that contains at least $sample.cov and $sample.nobs, potentially also $sample.mean, $sample.th, $NACOV, and $WLS.V. Also contains $lavOptions that will be passed to lavaan(...).

Note

The $lavOptions list will always set fixed.x=FALSE and conditional.x=FALSE. Users should not override those options when calling lavaan::lavaan() because doing so would yield incorrect *SE*s and test statistics. Computing the correct $NACOV argument would depend on which specific variables are treated as fixed, which would require an argument to poolSat() for users to declare names of exogenous variables. This has not yet been programmed, but that feature may be added in the future in order to reduce the number of parameters to estimate. However, if "exogenous" predictors were incomplete and imputed, then they are not truly fixed (i.e., unvarying across samples), so treating them as fixed would be illogical and yield biased *SE*s and test statistics.

The information returned by poolSat() must assume that any fitted SEM will include all the variables in $sample.cov and (more importantly) in $NACOV. Although lavaan can drop unused rows/columns from $sample.cov, it cannot be expected to drop the corresponding sampling variances of those eliminated (co)variances from $NACOV. Thus, it is necessary to use poolSat() to obtain the appropriate summary statistics for any particular SEM (see **Examples**).

Author(s)

Terrence D. Jorgensen (University of Amsterdam; <TJorgensen314@gmail.com>)

References

Lee, T., & Cai, L. (2012). Alternative multiple imputation inference for mean and covariance structure modeling. *Journal of Educational and Behavioral Statistics, 37*(6), 675–702. doi:10.3102/1076998612458320

Chung, S., & Cai, L. (2019). Alternative multiple imputation inference for categorical structural equation modeling, *Multivariate Behavioral Research, 54*(3), 323–337. doi:10.1080/00273171.2018.1523000

See Also

lavaan.mi() for traditional method (fit SEM to each imputation, pool results afterward).

Examples

```
data(HS20imps) # import a list of 20 imputed data sets

## fit saturated model to imputations, pool those results
impSubset1 <- lapply(HS20imps, "[", i = paste0("x", 1:9)) # only modeled variables
(prePooledData <- poolSat(impSubset1))

## Note: no means were returned (default lavOption() is meanstructure=FALSE)
(prePooledData <- poolSat(impSubset1, meanstructure = TRUE))

## specify CFA model from lavaan's ?cfa help page
HS.model <- '
  visual  =~ x1 + x2 + x3
  textual =~ x4 + x5 + x6
  speed   =~ x7 + x8 + x9
'

## fit model to summary statistics in "prePooledData"
```

```
fit <- cfa(HS.model, data = prePooledData, std.lv = TRUE)
## By default, the "Scaled" column provides a "scaled.shifted" test
## statistic that maintains an approximately nominal Type I error rate.
summary(fit, fit.measures = TRUE, standardized = "std.all")
## Note that this scaled statistic does NOT account for deviations from
## normality, because the default normal-theory standard errors were
## requested when running poolSat().  See below about non-normality.

## Alternatively, "Browne's residual-based (ADF) test" is also available:
lavTest(fit, test = "browne.residual.adf", output = "text")

## Optionally, save the saturated-model lavaan.mi object, which
## could be helpful for diagnosing convergence problems per imputation.
satFit <- poolSat(impSubset1, return.fit = TRUE)


## FITTING MODELS TO DIFFERENT (SUBSETS OF) VARIABLES

## If you only want to analyze a subset of these variables,
mod.vis <- 'visual  =~ x1 + x2 + x3'
## you will get an error:
try(
  fit.vis <- cfa(mod.vis, data = prePooledData) # error
)

## As explained in the "Note" section, you must use poolSat() again for
## this subset of variables
impSubset3 <- lapply(HS20imps, "[", i = paste0("x", 1:3)) # only modeled variables
visData <- poolSat(impSubset3)
fit.vis <- cfa(mod.vis, data = visData) # no problem


## OTHER lavaan OPTIONS


## fit saturated MULIPLE-GROUP model to imputations
impSubset2 <- lapply(HS20imps, "[", i = c(paste0("x", 1:9), "school"))
(prePooledData2 <- poolSat(impSubset2, group = "school",
                           ## request standard errors that are ROBUST
                           ## to violations of the normality assumption:
                           se = "robust.sem"))
## Nonnormality-robust standard errors are implicitly incorporated into the
## pooled weight matrix (NACOV= argument), so they are
## AUTOMATICALLY applied when fitting the model:
fit.config <- cfa(HS.model, data = prePooledData2, group = "school",
                  std.lv = TRUE)
## standard errors and chi-squared test of fit both robust to nonnormality
summary(fit.config)


## CATEGORICAL OUTCOMES

## discretize the imputed data, for an example of 3-category data
```

```
HS3cat <- lapply(impSubset1, function(x) {
  as.data.frame( lapply(x, cut, breaks = 3, labels = FALSE) )
})
## pool polychoric correlations and thresholds
(prePooledData3 <- poolSat(HS3cat, ordered = paste0("x", 1:9)))

fitc <- cfa(HS.model, data = prePooledData3, std.lv = TRUE)
summary(fitc)

## Optionally, use unweighted least-squares estimation.  However,
## you must first REMOVE the pooled weight matrix (WLS.V= argument)
## or replace it with an identity matrix of the same dimensions:
prePooledData4 <- prePooledData3
prePooledData4$WLS.V <- NULL
## or prePooledData4$WLS.V <- diag(nrow(prePooledData3$WLS.V))
fitcu <- cfa(HS.model, data = prePooledData4, std.lv = TRUE, estimator = "ULS")
## Note that the SEs and test were still appropriately corrected:
summary(fitcu)
```

`standardizedSolution.mi`

Standardized Pooled Parameter Estimates

Description

This function calculates pooled parameter estimates from a lavaan model fitted to multiple imputed data sets, then transforms the pooled estimates and their *SE*s using the delta method.

Usage

```
standardizedSolution.mi(
  object,
  return.vcov = FALSE,
  omit.imps = c("no.conv", "no.se"),
  ...
)
```

Arguments

`object`	An object of class `lavaan.mi`
`return.vcov`	`logical` indicating whether to return only the pooled asymptotic covariance matrix, `vcov(object)`, but transformed for standardized parameters. This is a way to obtain a pooled analog of `lavInspect(object, "vcov.std.all")` with a lavaan::lavaan object, and it is how the *SE*s are derived for standardized solutions.
`omit.imps`	`character` indicating criteria for excluding imputations from pooled results. See lavaan.mi for argument details.
`...`	Arguments passed to `lavaan::standardizedSolution()`.

Value

A `data.frame` containing standardized model parameters, analogous to `lavaan::standardizedSolution()`. Delta-method *SE*s and CIs rely on asymptotic theory, so only Wald *z* tests are available, analogous to setting `parameterEstimates.mi(fit, asymptotic = TRUE)`.

Author(s)

Terrence D. Jorgensen (University of Amsterdam; <TJorgensen314@gmail.com>)

See Also

`parameterEstimates.mi()` for pooling unstandardized parameter estimates, which can also add standardized point estimates to indicate effect size.

Examples

```
data(HS20imps) # import a list of 20 imputed data sets

## specify CFA model from lavaan's ?cfa help page
HS.model <- '
  visual  =~ x1 + x2 + x3
  textual =~ x4 + x5 + x6
  speed   =~ x7 + x8 + x9
'
## fit model to 20 imputed data sets
fit <- cfa.mi(HS.model, data = HS20imps)

standardizedSolution.mi(fit) # default: type = "std.all"

## only standardize latent variables:
standardizedSolution.mi(fit, type = "std.lv",
                        output = "text") # display like summary()
```

Index