Package 'piecewiseSEM'

December 8, 2016

Type Package

Title Piecewise Structural Equation Modeling

Version 1.2.1

Date 2016-12-06

Author Jon Lefcheck

Maintainer Jon Lefcheck <jslefche@vims.edu>

Description Implements piecewise structural equation models.

License GPL-3

Imports lavaan, lme4, methods, nlme, pbkrtest

Suggests knitr, rmarkdown

VignetteBuilder knitr

RoxygenNote 5.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2016-12-08 20:38:22

R topics documented:

`piecewiseSEM-package` *Piecewise Structural Equation Modeling*

Description

Implements piecewise structural equation modeling in R, complete with goodness-of-fit tests and retrieval of model coefficients. Compared with traditional variance-covariance based SEM, piecewise SEM allows for fitting of models to different distributions and/or incorporates hierarchical/nested random structures.

Supported model classes include: 'lm', 'rq', 'glm', 'glm.nb', 'gls', 'pgls', 'merMod', 'merModLmerTest', 'lme', 'glmmPQL', 'glmmadmb', and 'glmmTMB'.

Details

Package:	piecewiseSEM
Type:	Package
Version:	1.2.1
Date:	2016-12-06
Depends:	R (3.3), ggm, lavaan, nlme, pbkrtest
Suggests:	MASS, lme4
License:	MIT

The primary functions in the package are `sem.fit` which performs goodness-of-fit tests, and `sem.coefs` which returns path coefficients (standardized, if specified) and standard errors.

Author(s)

Jon Lefcheck <jslefche@vims.edu>

References

Shipley, Bill. "A new inferential test for path models based on directed acyclic graphs." Structural Equation Modeling 7.2 (2000): 206-218.

Shipley, Bill. Cause and correlation in biology: a user's guide to path analysis, structural equations and causal inference. Cambridge University Press, 2002.

Shipley, Bill. "Confirmatory path analysis in a generalized multilevel context." Ecology 90.2 (2009): 363-368.

Shipley, Bill. "The AIC model selection method applied to path analytic models compared using a d-separation test." Ecology 94.3 (2013): 560-564.

Examples

```
# Load example data
data(shipley2009)

# Reduce dataset for example
shipley2009.reduced = shipley2009[1:200, ]

# Load model packages
library(lme4)
library(nlme)

# Create list of models
  shipley2009.reduced.modlist = list(

    lme(DD ~ lat, random = ~1|site/tree, na.action = na.omit,
        data = shipley2009.reduced),

    lme(Date ~ DD, random = ~1|site/tree, na.action = na.omit,
        data = shipley2009.reduced),

    lme(Growth ~ Date, random = ~1|site/tree, na.action = na.omit,
        data = shipley2009.reduced),

    glmer(Live ~ Growth+(1|site)+(1|tree),
          family=binomial(link = "logit"), data = shipley2009.reduced)

  )

# Get goodness-of-fit and AIC
sem.fit(shipley2009.reduced.modlist, shipley2009.reduced)

# Extract path coefficients
sem.coefs(shipley2009.reduced.modlist, shipley2009.reduced)

## Not run:
  # Repeat with full dataset as in Shipley (2009)

  # Create list of models
  shipley2009.modlist = list(
```

```
    lme(DD ~ lat, random = ~1|site/tree, na.action = na.omit,
        data = shipley2009),

    lme(Date ~ DD, random = ~1|site/tree, na.action = na.omit,
        data = shipley2009),

    lme(Growth ~ Date, random = ~1|site/tree, na.action = na.omit,
        data = shipley2009),

    glmer(Live ~ Growth+(1|site)+(1|tree),
          family=binomial(link = "logit"), data = shipley2009)

  )

  # Get goodness-of-fit and AIC
  sem.fit(shipley2009.modlist, shipley2009)

  # Extract path coefficients
  sem.coefs(shipley2009.modlist, shipley2009)

## End(Not run)
```

`endogenous.reverse` *Updates basis set under certain conditions.*

Description

A helper function for `sem.missing.paths` that reverses independence claims between intermediate endogenous variables fit to non-normal distributions.

Usage

```
endogenous.reverse(basis.set, modelList, add.vars)
```

Arguments

`basis.set`	a `list` of tests of conditional independence claims obtained from `sem.basis.set`.
`modelList`	a `list` of regressions representing the structural equation model.
`add.vars`	a vector of additional variables whose independence claims should be evaluated, but which do not appear in the model list.

Value

Returns a `list` of conditional independence claims. The list contains duplicated entries, reversing the independence claims between endogenous variables that also predict other variables in the model *only if* those variables are fitted to a non-normal distribution.

This is a temporary workaround for GLM(M)s that do not adhere to the property of symmetricity of independence claims as a function of transformations via the link function.

Author(s)

Jon Lefcheck

Examples

```
## Not run:
  data = data.frame(
    a = runif(50),
    b = runif(50, 0, 1),
    c = runif(50, 0, 1),
    d = runif(50)
  )

  modelList = list(
    glm(b ~ a, family = "binomial", data),
    glm(c ~ a, family = "binomial", data),
    lm(d ~ b + c, data)
  )

  (basis.set = sem.basis.set(modelList))

  endogenous.reverse(basis.set, modelList)

## End(Not run)
```

---

filter.exogenous *Filter exogenous variables from the basis set for SEM*

Description

Identifies exogenous variables (variables that have no paths leading to them) and removes them from the basis set when they appear as responses.

Usage

```
filter.exogenous(modelList, basis.set, corr.errors, add.vars)
```

Arguments

modelList	a list of regressions representing the structural equation model.
basis.set	an list of vectors representing conditional independence statements.
corr.errors	a vector of variables with correlated errors (separated by "~~").
add.vars	a vector of additional variables whose independence claims should be evaluated, but which do not appear in the model list.

Details

Helper function for `sem.basis.set` that removes independence claims between two exogenous (i.e., independent) variables, as the direction of causality cannot be reliably inferred from the SEM model list.

Value

Returns a `list` of independence claims. Each entry in the `list` is a vector where the first entry is the predictor whose independence from the response is being evaluated, the second is the response, and remaining entries represent the variables on which the independence claim is conditional. E.g.,

[1] `"x1" "y" "x2"`

would be the missing path between `y ~ x1` conditional on `x2`. The basis set is constructed using the function `basiSet` in the `ggm` package.

Author(s)

Jon Lefcheck

References

Shipley, Bill. "A new inferential test for path models based on directed acyclic graphs." Structural Equation Modeling 7.2 (2000): 206-218.

`get.basis.set` *Generates basis set for tests of direct separation*

Description

A helper function for `sem.missing.paths` that returns a list of (conditional) independence claims from an adjacency matrix generated from `get.dag`.

Usage

```
get.basis.set(amat)
```

Arguments

`amat`	an adjacency matrix.

Value

Returns a list of independence claims. For each entry, the first argument is the variable whose partial regression slope should be zero, the second argument is the response, and the final entries are the variable(s) upon which the first argument is conditional.

Author(s)

Jon Lefcheck

get.dag — *Generates adjacency matrix from list of structured equations*

Description

A helper function for `sem.missing.paths` that generates an adjacency matrix from a list of model formulae (generally recovered from a list of structured equations).

Usage

```
get.dag(formulaList)
```

Arguments

`formulaList`	a list of model formula(e).

Value

Returns a `matrix` of 0s and 1s, 0 indicating no relationship and 1 indicating a relationship, with columns being predictors and rows being responses.

Author(s)

Jon Lefcheck

get.formula.list — *Generates list of formula from a list of models.*

Description

A helper function for `sem.missing.paths` that retrieves a formula list from a list of structured equations.

Usage

```
get.formula.list(modelList, add.vars)
```

Arguments

`modelList`	a `list` of regressions representing the structural equation model.
`add.vars`	a vector of additional variables whose independence claims should be evaluated, but which do not appear in the model list.

Value

Returns a `list` of objects of class `formula`.

Author(s)

Jon Lefcheck

`get.model.control` *Formats model control arguments*

Description

A helper function for `sem.missing.paths` and `sem.partial.resid` that formats a `list` of model control arguments to pass to `update`.

Usage

```
get.model.control(model, model.control)
```

Arguments

`model`	a linear model.
`model.control`	a `list` of control parameters corresponding to the model classes found in the SEM.

Value

Returns a `list` of model controls.

Author(s)

Jon Lefcheck

See Also

`glm.control`, `glsControl`, `lmeControl`, `lmerControl`, `glmerControl`

`get.random.formula` *Recovers random structure from a mixed model*

Description

A helper function for `sem.missing.paths` and `sem.partial.resid` that retrives (and modifies) the proper random structure for a mixed model to be passed to `update`.

Usage

```
get.random.formula(model, rhs,  modelList, dropterms)
```

Arguments

model	a linear model.
rhs	the fixed effects formula.
modelList	a list of regressions representing the structural equation model.
dropterms	terms to be dropped from the random structure (optional).

Value

Returns a character vector corresponding to the updated random structure.

Author(s)

Jon Lefcheck

See Also

findbars

---

get.scaled.data *Scales data frame*

Description

A helper function for sem.coefs that provides a scaled data frame (by mean and standard deviation, or range).

Usage

```
get.scaled.data(modelList, data, standardize)
```

Arguments

modelList	a list of regressions representing the structural equation model.
data	a data.frame used to construct the structured equations.
standardize	specifies the type of standardization to perform: "none", "scale", "range".

Details

Automatically calculates and scales variables that are transformed in the model formulae.

Value

Returns a data.frame with the scaled (transformed) variables.

Author(s)

Jon Lefcheck

See Also

scale

`get.scaled.model` *Scales data frame*

Description

A helper function for `sem.coefs` that updates a model object with a scaled data frame (by mean and standard deviation, or range).

Usage

```
get.scaled.model(model, newdata, modelList)
```

Arguments

`model`	a linear model.
`newdata`	a `data.frame` from `get.scaled.data`.
`modelList`	a `list` of regressions representing the structural equation model.

Value

Returns a model object fit to the new data.

Author(s)

Jon Lefcheck

See Also

scale

`get.sort.dag` *Sorts adjacency matrix from parents to children*

Description

A helper function for `sem.basis.set` that takes the adjacency matrix from `get.dag` and orders it based on the appearance of variables in the SEM. It first puts variables that have no links (i.e., exogenous variables), then those that have only links to exogenous variables, then links to both exogenous and those that have links (i.e., endogenous), and finally to only endogenous.

Usage

```
get.sort.dag(formulaList)
```

Arguments

formulaList	a list of model formula(e).

Value

Returns a matrix of 0s and 1s, 0 indicating no relationship and 1 indicating a relationship, with columns being predictors and rows being responses.

Author(s)

Jon Lefcheck

---

partial.resid *Calculates partial residuals for two variables*

Description

Extracts partial residuals for y ~ x | Z, where Z represents all other variables present in a structured equation upon which x is conditional.

Usage

```
partial.resid(.formula = y ~ x, modelList, data, model.control = NULL,
return.data.frame = TRUE, plotit = TRUE, plotreg = TRUE, plotCI = TRUE)
```

Arguments

.formula	a formula specifying the partial residuals to investigate.
modelList	a single model or list of regressions representing the structural equation model.
data	a data.frame used to construct the model.
model.control	a list of model control arguments to be passed to the partial residual models.
return.data.frame	whether a data.frame of the partial residuals should be returned. Default is TRUE.
plotit	whether the partial plot of y ~ x | Z should be returned. Default is TRUE.
plotreg	whether the partial regression of y ~ x | Z should also be plotted. Default is TRUE.
plotCI	whether the confidence intervals of the partial regression of y ~ x | Z should also be plotted. Default is TRUE.

Value

Returns a data.frame where the first column are the partial residuals of y ~ x | Z, and the second column is the partial residuals of x ~ Z | y.

If plotit = TRUE, then the function also returns a plot of y ~ x | Z.

If plotreg = TRUE, then the plot includes the linear regression of y ~ x | Z.

Author(s)

Jon Lefcheck

References

Shipley, Bill. Cause and correlation in biology: a user's guide to path analysis, structural equations and causal inference. Cambridge University Press, 2002.

Examples

```
# Load model package
library(nlme)

# Load data from Shipley (2013)
data(shipley2013)

shipley2013.modlist = list(

  lme(x2~x1, random = ~x1 | species, data = shipley2013),

  lme(x3~x2, random = ~x2 | species, data = shipley2013),

  lme(x4~x2, random = ~x2 | species, data = shipley2013),

  lme(x5~x3+x4, random = ~x3+x4 | species, data = shipley2013)

)

# Get partial residuals of x3 on x5 conditional on x4
resids.df = partial.resid(x5 ~ x3, shipley2013.modlist, shipley2013,
  list(lmeControl(opt = "optim")))

# Also returns raw residuals values for plotting in other packages
head(resids.df)

## Not run:
  # Create example data
  set.seed(1)

  example.data = data.frame(
    y = rnorm(100, 0, 1),
    x1 = rnorm(100, 10, 50),
    random = letters[1:5]
  )

  example.data$x2 = example.data$y + runif(100, 0, 5)

  example.data$x3 = example.data$x2 + runif(100, 0, 5)

  # Run regular linear model using lm()
  lm.model = lm(y ~ x1 + x2 + x3, example.data)
```

```
  partial.resid(y ~ x3, lm.model, example.data)

  # Works with interactions too
  lm.model2 = lm(y ~ x1 * x2 + x3, example.data)

  partial.resid(y ~ x1 * x2, lm.model2, example.data, return.data.frame = FALSE)

  # Run generalized least squared regression
  library(nlme)

  gls.model = gls(y ~ x1 + x2 + x3, example.data)

  partial.resid(y ~ x3, gls.model, example.data, return.data.frame = FALSE)

  # Run mixed effects model using lme()
  lme.model = lme(y ~ x1 + x2 + x3, random = ~ 1 | random, example.data)

  partial.resid(y ~ x3, lme.model, example.data, return.data.frame = FALSE)

  # Run mixed effects model using lmer()
  library(lme4)

  lmer.model = lmer(y ~ x1 + x2 + x3 + (1 | random), example.data)

  partial.resid(y ~ x3, lmer.model, example.data, return.data.frame = FALSE)

  # Remove some values of x3 from data.frame
  example.data[c(12, 23, 45), "x3"] = NA

  # Run regular linear model using lm()
  lm.model = lm(y ~ x1 + x2 + x3, example.data)

  sum(!is.na(partial.resid(y ~ x3, lm.model, example.data)$x.resids)) # Should be 97

## End(Not run)
```

`rsquared` *Goodness-of-fit statistics for linear models*

Description

Returns (pseudo)-R^2 and AIC values for component models in structural equation model (SEM).

Usage

```
rsquared(modelList, aicc = FALSE)
```

Arguments

modelList	a list of regressions representing the structural equation model.
aicc	whether AIC corrected for small sample size (AICc) should be returned. Default is FALSE.

Details

Returns goodness-of-fit statistics for generalized linear (mixed) models, including (marginal and condition) R^2 and Akaike Information Criterion (AIC(c)) values.

Value

Returns a data.frame with the model class, the family, the link function, the marginal R^2 (based on fixed effects only) and/or conditional R^2 (based on fixed and random effects, if present), and the AIC(c) score (based on ML).

Note

If the models in modelList are all fit the same response, the function automatically returns the delta AIC(c) as the final column.

This function sources the rsquared.glmm function found here: https://github.com/jslefche/rsquared.glmm, and thus may be periodically updated independently of this package.

Author(s)

Jon Lefcheck & Juan Casallas

References

Nakagawa, Shinichi, and Holger Schielzeth. "A general and simple method for obtaining R2 from generalized linear mixed-effects models." Methods in Ecology and Evolution 4.2 (2013): 133-142.

Johnson, Paul C.D. "Extension of Nakagawa & Schielzeth's R2GLMM to random slopes models." Methods in Ecology and Evolution.

Examples

```
# Load example data
data(shipley2009)

# Load model packages
library(lme4)
library(nlme)

# Create list of models
shipley2009.modlist = list(

  lme(DD ~ lat, random = ~1|site/tree, na.action = na.omit,
      data = shipley2009),
```

```
  lme(Date ~ DD, random = ~1|site/tree, na.action = na.omit,
      data = shipley2009),

  lme(Growth ~ Date, random = ~1|site/tree, na.action = na.omit,
      data = shipley2009),

  glmer(Live ~ Growth+(1|site)+(1|tree),
        family=binomial(link = "logit"), data = shipley2009)

)

# Return model fit statistics
rsquared(shipley2009.modlist)

## Not run:
  # Get R2 for linear model
  lm.mod = lm(DD ~ lat, data = shipley2009)
  rsquared(lm.mod)

  # Get R2 for generalized linear model
  glm.mod = glm(Live ~ Growth, family = "binomial", data = shipley2009)
  rsquared(glm.mod)

  # Get R2 for generalized least-squares model
  library(nlme)

  gls.mod = gls(DD ~ lat, na.action = na.omit, data = shipley2009)
  rsquared(gls.mod)

  # Can supply the models as a list
  # Use lm and gls -- should produce very similar R2s, will also produce delta AIC
  rsquared(list(lm.mod, gls.mod))

  # Get R2 for linear mixed effects model (nlme)
  lme.mod = lme(DD ~ lat, random = ~1|site/tree, na.action = na.omit, data = shipley2009)
  rsquared(lme.mod)

  # Get R2 for linear mixed effects model (lme4)
  library(lme4)

  lmer.mod = lmer(DD ~ lat + (1|site/tree), data = shipley2009)
  rsquared(lmer.mod)

  # Get R2 for generalized linear mixed effects model (lme4)
  glmer.mod = glmer(Live ~ Growth + (1|site/tree), family = "binomial", data = shipley2009)
  rsquared(glmer.mod)

## End(Not run)
```

---

sem.aic *Extracts AIC scores for piecewise SEM*

Description

Extracts the AIC and AICc (corrected for small sample size) values from a piecewise structural equation model (SEM).

Usage

```
sem.aic(modelList, data, corr.errors, add.vars, grouping.vars,
  grouping.fun, adjust.p, basis.set, pvalues.df, model.control,
  .progressBar)
```

Arguments

`modelList`	a `list` of regressions representing the structural equation model.
`data`	a `data.frame` used to construct the structured equations.
`corr.errors`	a vector of variables with correlated errors (separated by "~~").
`add.vars`	a vector of additional variables whose independence claims should be evaluated, but which do not appear in the model list.
`grouping.vars`	an optional variable that represents the levels of data aggregation for a multi-level dataset.
`grouping.fun`	a function defining how variables are aggregated in `grouping.vars`. Default is `mean`.
`adjust.p`	whether p-values degrees of freedom should be adjusted (see below). Default is `FALSE`.
`basis.set`	provide an optional basis set.
`pvalues.df`	an optional `data.frame` corresponding to p-values for independence claims.
`model.control`	a `list` of model control arguments to be passed to d-sep models.
`.progressBar`	enable optional text progress bar. Default is `TRUE`.

Details

This function calculates AIC and AICc (corrected for small sample sizes) values for a piecewise structural equation model (SEM).

For linear mixed effects models, p-values can be adjusted to accommodate the full model degrees of freedom using the argument `p.adjust = TRUE`. For more information, see Shipley 2013.

Value

Returns a `data.frame` where the first entry is the AIC score, and the second is the AICc score, and the third is the likelihood degrees of freedom (K).

Author(s)

Jon Lefcheck

References

Shipley, Bill. "The AIC model selection method applied to path analytic models compared using a d-separation test." Ecology 94.3 (2013): 560-564.

Examples

```
# Load example data
data(shipley2009)

# Reduce dataset for example
shipley2009.reduced = shipley2009[1:200, ]

# Load model packages
library(lme4)
library(nlme)

# Create list of models
  shipley2009.reduced.modlist = list(

    lme(DD ~ lat, random = ~1|site/tree, na.action = na.omit,
        data = shipley2009.reduced),

    lme(Date ~ DD, random = ~1|site/tree, na.action = na.omit,
        data = shipley2009.reduced),

    lme(Growth ~ Date, random = ~1|site/tree, na.action = na.omit,
        data = shipley2009.reduced),

    glmer(Live ~ Growth+(1|site)+(1|tree),
          family=binomial(link = "logit"), data = shipley2009.reduced)

  )

  # Get AIC and AICc values for the SEM
  sem.aic(shipley2009.reduced.modlist, shipley2009.reduced)

## Not run:
  # Repeat with full dataset as in Shipley (2009)

  # Create list of models
  shipley2009.modlist = list(

    lme(DD ~ lat, random = ~1|site/tree, na.action = na.omit,
        data = shipley2009),

    lme(Date ~ DD, random = ~1|site/tree, na.action = na.omit,
        data = shipley2009),

    lme(Growth ~ Date, random = ~1|site/tree, na.action = na.omit,
        data = shipley2009),

    glmer(Live ~ Growth+(1|site)+(1|tree),
```

```
            family=binomial(link = "logit"), data = shipley2009)

  )

  # Get AIC and AICc values for the SEM
  sem.aic(shipley2009.modlist, shipley2009)

## End(Not run)
```

`sem.basis.set` *Derive independence claims for SEM*

Description

Generates a list representing the (conditional) independence claims from a model list.

Usage

```
sem.basis.set(modelList, corr.errors, add.vars)
```

Arguments

`modelList`	a `list` of regressions representing the structural equation model.
`corr.errors`	a vector of variables with correlated errors (separated by `~~`).
`add.vars`	a vector of additional variables whose independence claims should be evaluated, but which do not appear in the model list.

Details

Variables with correlated errors have no direct relationship but rather are hypothesized to be driven by the same underlying factor. This covariance should be reflected as correlated errors (double-headed arrow). Correlated errors are specified using the same syntax as the `lavaan` package: `var1 ~~ var2`. Variables with correlated errors are ignored in the basis set under the assumption that their correlations will be quantified later using the function `sem.coefs`.

The argument `add.vars` requires a vector of character strings corresponding to column names in the dataset used to construct the models in `modelList`. This is useful if comparing nested SEMs where one wishes to account for additional variables whose independence claims should be evaluated, but which do not have any hypothesized paths in the current SEM. The default assumes there is no additional independence claims that do not appear in the model list.

Value

Returns a `list` of independence claims. Each entry in the `list` is a vector where the first entry is the predictor whose independence from the response is being evaluated, the second is the response, and remaining entries represent the variables on which the independence claim is conditional. E.g.,

[1] `"x1" "y" "x2"`

would be the missing path between `y ~ x1` conditional on `x2`. The basis set is constructed using the function `basiSet` in the `ggm` package.

Note

Unlike the functions DAG and basiSet in the ggm package, this function incorporates interactions if they included in the models.

Author(s)

Jon Lefcheck

References

Shipley, Bill. "A new inferential test for path models based on directed acyclic graphs." Structural Equation Modeling 7.2 (2000): 206-218.

Examples

```
# Load example data
data(shipley2009)

# Load model packages
library(lme4)
library(nlme)

# Create list of models
shipley2009.modlist = list(

  lme(DD ~ lat, random = ~1|site/tree, na.action = na.omit,
      data = shipley2009),

  lme(Date ~ DD, random = ~1|site/tree, na.action = na.omit,
      data = shipley2009),

  lme(Growth ~ Date, random = ~1|site/tree, na.action = na.omit,
      data = shipley2009),

  glmer(Live ~ Growth+(1|site)+(1|tree),
        family=binomial(link = "logit"), data = shipley2009)

)

# Get AIC and AICc values for the SEM
sem.basis.set(shipley2009.modlist)
```

---

sem.coefs *Retrieves path coefficients for structural equation model*

Description

Extracts path coefficients for a piecewise structural equation model (SEM).

Usage

```
sem.coefs(modelList, data, standardize = "none", corr.errors, intercept = FALSE)
```

Arguments

modelList	a list of regressions representing the structural equation model.
data	a data.frame used to construct the structured equations.
standardize	specifies the type of standardization to perform: "none", "scale", "range".
corr.errors	a vector of variables with correlated errors (separated by ~~).
intercept	whether values and significance tests for the intercept should be returned; default is FALSE.

Details

Unstandardized coefficients are extracted directly from model output. For models of class lmerMod, denominator degrees of freedom and resulting P-values are calculated using the Kenward-Rogers approximation from the pbkrtest package.

When standardize = "scale", variables are scaled by mean and variance using the function scale. When standardize = "range", variables are scaled by subtracting the minimum and dividing by the difference of the range.

If predictors are factors or dummy variables, then they are left in their original units. If the response is not normally distributed, then it is left untransformed even when standardize = "scale" or standardize = "range". Predictors in these models are transformed, unless they meet the previous criteria.

Variables with correlated errors have no direct relationship but rather are hypothesized to be driven by the same underlying factor. This covariance should be reflected as correlated errors (double-headed arrow in a path diagram). Correlated errors are specified using the syntax from the lavaan package: var1 ~~ var2. If two exogenous variables are listed, then the partial Pearson correlation between the two is given, otherwise the correlation between the partial residuals is returned using partial.resid. Significance tests (i.e., P-values) for correlated errors are derived from a t-distribution using the transformation in Shipley, 2000 (p.76).

Value

Returns a data.frame with the causal path (response, predictor), the coefficient estimate, the standard error, and associated p-value.

Note

Caution: There is some debate over whether coefficients and significance tests for interactions containing standardized variables are valid (see Bollen 1989). This usually refers to the fact that some programs standardize the product, instead of the component variables. piecewiseSEM standardizes variables before calculating their product and thus should produce valid coefficients. The standard errors on the coefficients are still a grey area: if you are concerned at all, refer to the SEs and P-values of the /un/standardized variables.

Author(s)

Jon Lefcheck

References

Shipley, Bill. Cause and correlation in biology: a user's guide to path analysis, structural equations and causal inference. Cambridge University Press, 2002.

Bollen, K. A. Structural equations with latent variables. New York: John Wiley & Sons, 1989.

See Also

scale

Examples

```
# Load example data
data(shipley2009)

# Load model packages
library(lme4)
library(nlme)

# Create list of models
shipley2009.modlist = list(

  lme(DD ~ lat, random = ~1|site/tree, na.action = na.omit,
      data = shipley2009),

  lme(Date ~ DD, random = ~1|site/tree, na.action = na.omit,
      data = shipley2009),

  lme(Growth ~ Date, random = ~1|site/tree, na.action = na.omit,
      data = shipley2009),

  glmer(Live ~ Growth+(1|site)+(1|tree),
        family=binomial(link = "logit"), data = shipley2009)

)

# Get raw path coefficients
sem.coefs(shipley2009.modlist, shipley2009)

# Scale by mean and variance
sem.coefs(shipley2009.modlist, shipley2009, standardize = "scale")

# Scale by range
sem.coefs(shipley2009.modlist, shipley2009, standardize = "range")
```

`sem.fisher.c` *Goodness-of-fit test for piecewise SEM*

Description

Evaluates independence claims for a piecewise structural equation model (SEM), and calculates Fisher's C and associated p-value to evaluate model fit.

Usage

```
sem.fisher.c(modelList, data, corr.errors, add.vars, grouping.vars, grouping.fun,
  adjust.p, basis.set, pvalues.df, model.control, .progressBar)
```

Arguments

`modelList`	a `list` of regressions representing the structural equation model.
`data`	a `data.frame` used to construct the structured equations.
`corr.errors`	a vector of variables with correlated errors (separated by "~~").
`add.vars`	a vector of additional variables whose independence claims should be evaluated, but which do not appear in the model list.
`grouping.vars`	an optional variable that represents the levels of data aggregation for a multi-level dataset.
`grouping.fun`	a function defining how variables are aggregated in `grouping.vars`. Default is `mean`.
`adjust.p`	whether p-values degrees of freedom should be adjusted (see below). Default is `FALSE`.
`basis.set`	provide an optional basis set.
`pvalues.df`	an optional `data.frame` corresponding to p-values for independence claims.
`model.control`	a `list` of model control arguments to be passed to d-sep models.
`.progressBar`	enable optional text progress bar. Default is `TRUE`.

Value

Returns a `data.frame` with the first entry corresponding to Fisher's C statistic, the second corresponding to the Chi-squared test degrees of freedom, and the third corresponding to the outcome (p-value) of the significance test derived from a Chi-squared distribution.

For linear mixed effects models, p-values can be adjusted to accommodate the full model degrees of freedom using the argument `p.adjust = TRUE`. For more information, see Shipley 2013.

Author(s)

Jon Lefcheck

References

Shipley, Bill. "Confirmatory path analysis in a generalized multilevel context." Ecology 90.2 (2009): 363-368.

Examples

```
# Load example data
data(shipley2009)

# Reduce dataset for example
shipley2009.reduced = shipley2009[1:200, ]

# Load model packages
library(lme4)
library(nlme)

# Create list of models
  shipley2009.reduced.modlist = list(

    lme(DD ~ lat, random = ~1|site/tree, na.action = na.omit,
        data = shipley2009.reduced),

    lme(Date ~ DD, random = ~1|site/tree, na.action = na.omit,
        data = shipley2009.reduced),

    lme(Growth ~ Date, random = ~1|site/tree, na.action = na.omit,
        data = shipley2009.reduced),

    glmer(Live ~ Growth+(1|site)+(1|tree),
          family=binomial(link = "logit"), data = shipley2009.reduced)

  )

# Get goodness-of-fit statistics
sem.fisher.c(shipley2009.reduced.modlist, shipley2009.reduced)

## Not run:
  # Repeat with full dataset as in Shipley (2009)

  # Create list of models
  shipley2009.modlist = list(

    lme(DD ~ lat, random = ~1|site/tree, na.action = na.omit,
        data = shipley2009),

    lme(Date ~ DD, random = ~1|site/tree, na.action = na.omit,
        data = shipley2009),

    lme(Growth ~ Date, random = ~1|site/tree, na.action = na.omit,
        data = shipley2009),

    glmer(Live ~ Growth+(1|site)+(1|tree),
```

```
            family=binomial(link = "logit"), data = shipley2009)

  )

  # Get goodness-of-fit statistics
  sem.fisher.c(shipley2009.modlist, shipley2009)

## End(Not run)
```

`sem.fit` *Goodness-of-fit tests for piecewise SEM*

Description

Tests independence claims and calculates Fisher's C statistic and associated p-value, and AIC and AICc, for a piecewise structural equation model (SEM).

Usage

```
sem.fit(modelList, data, conditional = FALSE, corr.errors = NULL, add.vars = NULL,
  grouping.vars = NULL, grouping.fun = mean, adjust.p = FALSE, basis.set = NULL,
  pvalues.df = NULL, model.control = NULL, .progressBar = TRUE)
```

Arguments

`modelList`	a `list` of regressions representing the structural equation model.
`data`	a `data.frame` used to construct the structured equations.
`conditional`	whether conditional variables should be shown in the independence claim (unless the formula is fewer than 30 characters). Default is FALSE.
`corr.errors`	a vector of variables with correlated errors (separated by "~~").
`add.vars`	a vector of additional variables whose independence claims should be evaluated, but which do not appear in the model list.
`grouping.vars`	an optional variable that represents the levels of data aggregation for a multilevel dataset.
`grouping.fun`	a function defining how variables are aggregated in `grouping.vars`. Default is `mean`.
`adjust.p`	whether p-values degrees of freedom should be adjusted (see below). Default is `FALSE`.
`basis.set`	provide an optional basis set.
`pvalues.df`	an optional `data.frame` corresponding to p-values for independence claims.
`model.control`	a `list` of model control arguments to be passed to d-sep models.
`.progressBar`	enable optional text progress bar. Default is `TRUE`.

Details

Independence claims are constructed based on how the variables are treated as in the model list. For example, if the indepedence claim includes a binary variable that is fit to a binomial distribution using an identity link, the function will evaluate the any claims using the same parameters.

Similarly, for linear mixed effects models construted in `lme4` or `nlme`, varying slopes and intercepts are treated as in the model list. For example, if a variable is modeled with both a random slope and intercept in any model in the model list, that variable will be modeled with a random slope and intercept when evaluating all independence claims in which it appears. If slopes and intercepts vary for multiple variables, they will appear as such, even if they are conditional.

For models of class `lmerMod`, denominator degrees of freedom and resulting P-values are calculated using the Kenward-Rogers approximation from the `pbkrtest` package.

For linear mixed effects models, p-values can be adjusted to accommodate the full model degrees of freedom using the argument `p.adjust = TRUE`. For more information, see Shipley 2013.

Variables with correlated errors have no direct relationship but rather are hypothesized to be driven by the same underlying factor. This covariance should be reflected as correlated errors (double-headed arrow). Correlated errors are specified using the syntax from the `lavaan` package: `var1 ~~ var2`. Variables with correlated errors are ignored in the basis set under the assumption that their correlations will be quantified later using the function `sem.coefs`.

The argument `add.vars` requires a vector of character strings corresponding to column names in the dataset used to construct the models in `modelList`. This is useful if comparing nested SEMs where one wishes to account for additional variables whose independence claims should be evaluated, but which do not have any hypothesized paths in the current SEM. The default assumes there is no additional independence claims that do not appear in the model list.

If the data is hierarchical dataset and one or more responses are identical for each level of a grouping factors – artificially inflating the degrees of freedom – users can summarize the dataset for each grouping factor(s) specified in the argument `grouping.vars`. For example, consider a two-level hierarchy, where variables at the top level have identical values for each level of the grouping variable. If the response is a top level variable (is identically replicated for the grouping variable), this function takes a mean of the lower level variables for each level of the grouping variable, then runs the test of d-separation. If the response is fully replicated (occurs at the lower level), then no aggregation occurs.

Value

Returns a `list` with the following:

`missing.paths`	A `data.frame` where the first column is the independence claim, and the second through sixth columns the model estimates corresponding to the response variable in the independence claim.
`fishers.c`	A `data.frame` with the first entry corresponding to Fisher's C statistic, the second corresponding to the Chi-squared test degrees of freedom, and the third corresponding to the outcome (p-value) of the significance test derived from a Chi-squared distribution.
`AIC`	A `data.frame` where the first entry is the AIC score, and the second is the AICc score, and the third is the likelihood degrees of freedom (K).

Note

The model controls in `model.control` will be assigned to any subsequent d-sep models derived from that model class.

Author(s)

Jon Lefcheck

References

Shipley, Bill. "Confirmatory path analysis in a generalized multilevel context." Ecology 90.2 (2009): 363-368.

Shipley, Bill. "The AIC model selection method applied to path analytic models compared using a d-separation test." Ecology 94.3 (2013): 560-564.

See Also

DAG, sem.missing.paths, sem.fisher.c, AIC, get_ddf_Lb

Examples

```
# Load example data
data(shipley2009)

# Reduce dataset for example
shipley2009.reduced = shipley2009[1:200, ]

# Load model packages
library(lme4)
library(nlme)

# Create list of models
  shipley2009.reduced.modlist = list(

    lme(DD ~ lat, random = ~1|site/tree, na.action = na.omit,
        data = shipley2009.reduced),

    lme(Date ~ DD, random = ~1|site/tree, na.action = na.omit,
        data = shipley2009.reduced),

    lme(Growth ~ Date, random = ~1|site/tree, na.action = na.omit,
        data = shipley2009.reduced),

    glmer(Live ~ Growth+(1|site)+(1|tree),
          family=binomial(link = "logit"), data = shipley2009.reduced)

  )

# Conduct goodness-of-fit tests
sem.fit(shipley2009.reduced.modlist, shipley2009.reduced)
```

```
## Not run:
  # Repeat with full dataset as in Shipley (2009)

  # Create list of models
  shipley2009.modlist = list(

    lme(DD ~ lat, random = ~1|site/tree, na.action = na.omit,
        data = shipley2009),

    lme(Date ~ DD, random = ~1|site/tree, na.action = na.omit,
        data = shipley2009),

    lme(Growth ~ Date, random = ~1|site/tree, na.action = na.omit,
        data = shipley2009),

    glmer(Live ~ Growth+(1|site)+(1|tree),
          family=binomial(link = "logit"), data = shipley2009)

  )

  # Conduct goodness-of-fit tests
  sem.fit(shipley2009.modlist, shipley2009)

  ###

  # Repeat but use lme4 to construct all models
  shipley2009.modlist.lme4 = list(

    lmer(DD~lat + (1|site/tree), na.action = na.omit,
         data = shipley2009),

    lmer(Date~DD + (1|site/tree), na.action = na.omit,
         data = shipley2009),

    lmer(Growth~Date + (1|site/tree), na.action = na.omit,
         data = shipley2009),

    glmer(Live~Growth+(1|site)+(1|tree),
          family=binomial(link = "logit"), data = shipley2009)

  )

  sem.fit(shipley2009.modlist.lme4, shipley2009)

  # Add new variable with no hypothesized links
  set.seed(1)
  add.var = rnorm(nrow(shipley2009), 50, 20)

  # Test for independence
  sem.fit(shipley2009.modlist, shipley2009, add.vars = c("add.var"))

  ###
```

```
# Add new variable grouped at site level
set.seed(2)

shipley2009$site.var = rep(rnorm(20, 10, 5), each = 95)

# Add a final model regressing at the level of the grouping variable
shipley2009.modlist2 = append(

  shipley2009.modlist, list(

   lm(site.var~Growth, data = aggregate(shipley2009, by = list(site = shipley2009[,"site"]),
                                        mean, na.rm = TRUE)))

)

# Test for independence
sem.fit(shipley2009.modlist2, shipley2009, grouping.vars = c("site"))

###

# Fit model from Shipley (2013)
data(shipley2013)

shipley2013.modlist = list(

  lme(x2~x1, random = ~x1 | species, data = shipley2013),

  lme(x3~x2, random = ~x2 | species, data = shipley2013),

  lme(x4~x2, random = ~x2 | species, data = shipley2013),

  lme(x5~x3+x4, random = ~x3+x4 | species, data = shipley2013)

)

sem.fit(shipley2013.modlist, shipley2013) # Convergence error!

# Specify model controls by switching to old optimizer
sem.fit(shipley2013.modlist, shipley2013, model.control = list(lmeControl(opt = "optim")))

###

# Compare to output from `ggm` package

# Load library
library(ggm)

# Load data
data(marks)
# Generate direct acyclic graph (DAG)
dag = DAG(mechanics ~ vectors+algebra,
          vectors ~ algebra,
          statistics ~ algebra+analysis,
```

```
          analysis ~ algebra)
  # Run test of directed separation
  shipley.test(dag, cov(marks), n=88)

  # Now create list of structured equations using lm()
  modelList = list(
    lm(mechanics ~ vectors+algebra, data = marks),
    lm( vectors ~ algebra, data = marks),
    lm(statistics ~ algebra+analysis, data = marks),
    lm(analysis ~ algebra, data = marks)
  )

  # Test model fit
  sem.fisher.c(modelList, marks)

## End(Not run)
```

`sem.lavaan` *Constructs variance-covariance based SEM*

Description

Estimates variance-covariance based (traditional) structural equation model (SEM) using the `lavaan` package.

Usage

```
sem.lavaan(modelList, data, compute.int = TRUE, corr.errors, add.vars, ...)
```

Arguments

`modelList`	a list of regressions representing the structural equation model.
`data`	a `data.frame` containing the full dataset for the SEM.
`compute.int`	a logical argument whether the interactions should be computed by hand. Default is `TRUE`.
`corr.errors`	a vector of variables with correlated errors (separated by "~~").
`add.vars`	a vector of additional variables whose independence claims should be evaluated, but which do not appear in the model list.
`...`	additional arguments passed to `lavaan`.

Details

Interactions among numeric variables are computed by hand, appended to the original `data.frame`, and modified in the formulae passed to `lavaan::sem`.

Value

Returns a model object of class `lavaan`.

Note

Models fit with `lm` should return identical coefficients to those generated using `sem.lavaan`.

Author(s)

Jon Lefcheck

References

Grace, James B. Structural equation modeling and natural systems. Cambridge University Press, 2006.

See Also

`sem`

Examples

```
# Load example data
data(shipley2009)

# Load model packages
library(lme4)
library(nlme)

# Create list of models
shipley2009.modlist = list(

  lme(DD ~ lat, random = ~1|site/tree, na.action = na.omit,
      data = shipley2009),

  lme(Date ~ DD, random = ~1|site/tree, na.action = na.omit,
      data = shipley2009),

  lme(Growth ~ Date, random = ~1|site/tree, na.action = na.omit,
      data = shipley2009),

  glmer(Live ~ Growth+(1|site)+(1|tree),
        family=binomial(link = "logit"), data = shipley2009)

)

# Get variance-covariance based SEM
(lavaan.model = sem.lavaan(shipley2009.modlist, shipley2009))

lavaan::summary(lavaan.model)

## Not run:
  # Load data from Shipley (2013)
  data(shipley2013)

  shipley2013.modlist = list(
```

```
    lme(x2~x1, random = ~x1 | species, data = shipley2013),

    lme(x3~x2, random = ~x2 | species, data = shipley2013),

    lme(x4~x2, random = ~x2 | species, data = shipley2013),

    lme(x5~x3+x4, random = ~x3+x4 | species, data = shipley2013)

  )

  # Get variance-covariance based SEM
  lavaan::summary(sem.lavaan(shipley2013.modlist, shipley2013))

## End(Not run)
```

`sem.missing.paths` *Evaluate independence claims for piecewise SEM*

Description

Identifies missing paths from a piecewise SEM, fits models, extracts path p-values and returns in a `data.frame`.

Usage

```
sem.missing.paths(modelList, data, conditional = FALSE, corr.errors = NULL,
add.vars = NULL, grouping.vars = NULL, grouping.fun = mean, adjust.p = FALSE,
basis.set, model.control = NULL, .progressBar = TRUE)
```

Arguments

`modelList`	a `list` of regressions representing the structural equation model.
`data`	a `data.frame` used to construct the structured equations.
`conditional`	whether conditional variables should be shown in the independence claim (unless the formula is fewer than 30 characters). Default is `FALSE`.
`corr.errors`	a vector of variables with correlated errors (separated by "~~").
`add.vars`	a vector of additional variables whose independence claims should be evaluated, but which do not appear in the model list.
`grouping.vars`	an optional variable that represents the levels of data aggregation for a multilevel dataset.
`grouping.fun`	a function defining how variables are aggregated in `grouping.vars`. Default is `mean`.
`adjust.p`	whether p-values degrees of freedom should be adjusted (see below). Default is `FALSE`.
`basis.set`	provide an optional basis set.
`model.control`	a `list` of model control arguments to be passed to d-sep models.
`.progressBar`	enable optional text progress bar. Default is `TRUE`.

Details

This function takes a model list (and optional basis set) and evaluates all conditional independence claims by constructing regressions, returning the claims, the variables upon which they are conditional, and associated p-values in a `data.frame`.

Value

Returns a `data.frame` where the first column is the independence claim (with the first variable being the variable of interest, followed by the conditional variables, unless truncated), and the second through sixth columns the model estimates corresponding to the response variable in the independence claim.

Note

Independence claims are constructed based on how the variables are treated as in the model list. For example, if the indepedence claim includes a binary variable that is fit to a binomial distribution using an identity link, the function will evaluate the any claims using the same parameters.

Similarly, for linear mixed effects models construted in `lme4` or `nlme`, varying slopes and intercepts are treated as in the model list. For example, if a variable is modeled with both a random slope and intercept in any model in the model list, that variable will be modeled with a random slope and intercept when evaluating all independence claims in which it appears. If slopes and intercepts vary for multiple variables, they will appear as such, even if they are conditional.

For models of class `lmerMod`, denominator degrees of freedom and resulting P-values are calculated using the Kenward-Rogers approximation from the `pbkrtest` package.

For linear mixed effects models, p-values can be adjusted to accommodate the full model degrees of freedom using the argument `p.adjust = TRUE`. For more information, see Shipley 2013.

Author(s)

Jon Lefcheck

References

Shipley, Bill. "Confirmatory path analysis in a generalized multilevel context." Ecology 90.2 (2009): 363-368.

Shipley, Bill. "The AIC model selection method applied to path analytic models compared using a d-separation test." Ecology 94.3 (2013): 560-564.

See Also

`DAG`, `get_ddf_Lb`

Examples

```
# Load example data
data(shipley2009)

# Reduce dataset for example
```

```
shipley2009.reduced = shipley2009[1:200, ]

# Load model packages
library(lme4)
library(nlme)

# Create list of models
  shipley2009.reduced.modlist = list(

    lme(DD ~ lat, random = ~1|site/tree, na.action = na.omit,
        data = shipley2009.reduced),

    lme(Date ~ DD, random = ~1|site/tree, na.action = na.omit,
        data = shipley2009.reduced),

    lme(Growth ~ Date, random = ~1|site/tree, na.action = na.omit,
        data = shipley2009.reduced),

    glmer(Live ~ Growth+(1|site)+(1|tree),
          family=binomial(link = "logit"), data = shipley2009.reduced)

  )

# Evaluate independence claims
sem.missing.paths(shipley2009.reduced.modlist, shipley2009.reduced)

## Not run:
  # Repeat with full dataset as in Shipley (2009)

  # Create list of models
  shipley2009.modlist = list(

    lme(DD ~ lat, random = ~1|site/tree, na.action = na.omit,
        data = shipley2009),

    lme(Date ~ DD, random = ~1|site/tree, na.action = na.omit,
        data = shipley2009),

    lme(Growth ~ Date, random = ~1|site/tree, na.action = na.omit,
        data = shipley2009),

    glmer(Live ~ Growth+(1|site)+(1|tree),
          family=binomial(link = "logit"), data = shipley2009)

  )

  # Evaluate independence claims
  sem.missing.paths(shipley2009.modlist, shipley2009)

## End(Not run)
```

`sem.model.fits` *Goodness-of-fit statistics for linear models*

Description

Returns (pseudo)-R^2 and AIC values for component models in structural equation model (SEM).

Usage

```
sem.model.fits(modelList, aicc = FALSE)
```

Arguments

`modelList`	a `list` of regressions representing the structural equation model.
`aicc`	whether AIC corrected for small sample size (AICc) should be returned. Default is `FALSE`.

Details

Returns goodness-of-fit statistics for generalized linear (mixed) models, including (marginal and condition) R^2 and Akaike Information Criterion (AIC(c)) values.

Value

Returns a `data.frame` with the model class, the family, the link function, the marginal R^2 (based on fixed effects only) and/or conditional R^2 (based on fixed and random effects, if present), and the AIC(c) score (based on ML).

Note

If the models in `modelList` are all fit the same response, the function automatically returns the delta AIC(c) as the final column.

This function sources the `rsquared.glmm` function found here: https://github.com/jslefche/rsquared.glmm, and thus may be periodically updated independently of this package.

Author(s)

Jon Lefcheck & Juan Casallas

References

Nakagawa, Shinichi, and Holger Schielzeth. "A general and simple method for obtaining R2 from generalized linear mixed-effects models." Methods in Ecology and Evolution 4.2 (2013): 133-142.

Johnson, Paul C.D. "Extension of Nakagawa & Schielzeth's R2GLMM to random slopes models." Methods in Ecology and Evolution.

Examples

```
# Load example data
data(shipley2009)

# Load model packages
library(lme4)
library(nlme)

# Create list of models
shipley2009.modlist = list(

  lme(DD ~ lat, random = ~1|site/tree, na.action = na.omit,
      data = shipley2009),

  lme(Date ~ DD, random = ~1|site/tree, na.action = na.omit,
      data = shipley2009),

  lme(Growth ~ Date, random = ~1|site/tree, na.action = na.omit,
      data = shipley2009),

  glmer(Live ~ Growth+(1|site)+(1|tree),
        family=binomial(link = "logit"), data = shipley2009)

)

# Return model fit statistics
sem.model.fits(shipley2009.modlist)

## Not run:
  # Get R2 for linear model
  lm.mod = lm(DD ~ lat, data = shipley2009)
  sem.model.fits(lm.mod)

  # Get R2 for generalized linear model
  glm.mod = glm(Live ~ Growth, family = "binomial", data = shipley2009)
  sem.model.fits(glm.mod)

  # Get R2 for generalized least-squares model
  library(nlme)

  gls.mod = gls(DD ~ lat, na.action = na.omit, data = shipley2009)
  sem.model.fits(gls.mod)

  # Can supply the models as a list
  # Use lm and gls -- should produce very similar R2s, will also produce delta AIC
  sem.model.fits(list(lm.mod, gls.mod))

  # Get R2 for linear mixed effects model (nlme)
  lme.mod = lme(DD ~ lat, random = ~1|site/tree, na.action = na.omit, data = shipley2009)
  sem.model.fits(lme.mod)

  # Get R2 for linear mixed effects model (lme4)
```

```
  library(lme4)

  lmer.mod = lmer(DD ~ lat + (1|site/tree), data = shipley2009)
  sem.model.fits(lmer.mod)

  # Get R2 for generalized linear mixed effects model (lme4)
 glmer.mod = glmer(Live ~ Growth + (1|site/tree), family = "binomial", data = shipley2009)
  sem.model.fits(glmer.mod)

## End(Not run)
```

`sem.plot` *Plotting of piecewise SEMs*

Description

Generates mandala plot for piecewise structural equation models.

Usage

```
sem.plot(modelList, data, coef.table, corr.errors = NULL,
show.nonsig = TRUE, scaling = 10, alpha = 0.05, ...)
```

Arguments

`modelList`	a `list` of regressions representing the structural equation model.
`data`	a `data.frame` used to construct the structured equations.
`coef.table`	a table of model coefficients obtained from `sem.coefs`.
`corr.errors`	a vector of variables with correlated errors (separated by ~~).
`show.nonsig`	whether to show non-significant paths (P >= alpha). Default is `TRUE`.
`scaling`	scaling coefficient for path thickness. Default is 10. If `NA` then all paths are equal width.
`alpha`	significance threshold. Default is 0.05.
`...`	additional arguments to `sem.coefs`.

Details

Accepts either a list of structured equations and the data, or a table of coefficients obtained from `sem.coefs`.

Value

Returns a plot where variables are equidistantly spaced in a circle. Arrows indicate the directionality of the relationship. The arrow width indicates the strength of the effect (based on estimates from `sem.coefs`). Dashed and grey arrows indicate non-significant effects. Red arrows indicate negative effects.

Author(s)

Jon Lefcheck

See Also

sem.coefs

Examples

```
# Load example data
data(shipley2009)

# Load model packages
library(lme4)
library(nlme)

# Create list of models
shipley2009.modlist = list(

  lme(DD ~ lat, random = ~1|site/tree, na.action = na.omit,
      data = shipley2009),

  lme(Date ~ DD, random = ~1|site/tree, na.action = na.omit,
      data = shipley2009),

  lme(Growth ~ Date, random = ~1|site/tree, na.action = na.omit,
      data = shipley2009),

  glmer(Live ~ Growth+(1|site)+(1|tree),
        family=binomial(link = "logit"), data = shipley2009)

)

# Plot output
sem.plot(shipley2009.modlist, shipley2009)

## Not run:
  # Alternately get coefficient table and pass to sem.plot
  coef.table = sem.coefs(shipley2009.modlist, shipley2009)

  sem.plot(coef.table)

## End(Not run)
```

---

sem.predict *Returns model predictions for SEM*

Description

Returns predictions for responses in the list of structured equations.

Usage

```
sem.predict(object, newdata, sefit = FALSE, ...)
```

Arguments

object	a single model or list of regressions representing the structural equation model.
newdata	a data.frame of predictors used to generate the model predictions.
sefit	whether standard errors of predictions should be returned. Default is FALSE.
...	additional arguments passed to predict.

Details

Mixed model predictions includes only fixed effects and *not* random effects (default is level = 0 or re.form = 0). This can be changed by passing additional arguments from predict.lme and predict.merMod.

If sefit = TRUE for mixed models, then standard errors on predictions are estimated using fixed effects *only*. See explanation here: http://glmm.wikidot.com/faq.

Value

Returns a data.frame containing the new data and the predicted responses based on fixed effects only for the innermost level of grouping (i.e., population residuals) corresponding to re.form = NA for lme4 models and level = 0 for nlme models.

Author(s)

Jon Lefcheck

See Also

predict, predict.lme, predict.merMod

Examples

```
# Load example data
data(shipley2009)

# Load model packages
library(lme4)
library(nlme)

# Create list of models
shipley2009.modlist = list(

  lme(DD ~ lat, random = ~1|site/tree, na.action = na.omit,
      data = shipley2009),

  lme(Date ~ DD, random = ~1|site/tree, na.action = na.omit,
      data = shipley2009),
```

```
  lme(Growth ~ Date, random = ~1|site/tree, na.action = na.omit,
      data = shipley2009),

  glmer(Live ~ Growth+(1|site)+(1|tree),
        family=binomial(link = "logit"), data = shipley2009)

)

# Create new data for predictions
shipley2009.new = data.frame(
  DD = seq(min(shipley2009$DD, na.rm = TRUE),
           max(shipley2009$DD, na.rm = TRUE),
           by = 0.01)
)

# Generate predictions
shipley2009.new.pred = sem.predict(shipley2009.modlist, shipley2009.new)
head(shipley2009.new.pred)

# Plot predicted fit
plot(shipley2009$Date ~ shipley2009$DD, col = "grey60")
lines(shipley2009.new.pred$Date.fit ~ shipley2009.new.pred$DD, lwd = 2, col = "red")

# Generate predictions with standard errors (based on fixed effects only)
shipley2009.new.pred = sem.predict(shipley2009.modlist, shipley2009.new, sefit = TRUE)

# Add 95 percent confidence intervals
lines(shipley2009.new.pred$DD,
      shipley2009.new.pred$Date.fit + 2 * shipley2009.new.pred$Date.se.fit,
      lwd = 1.5, lty = 2, col = "red")

lines(shipley2009.new.pred$DD,
      shipley2009.new.pred$Date.fit - 2 * shipley2009.new.pred$Date.se.fit,
      lwd = 1.5, lty = 2, col = "red")
```

`shipley2009` *Shipley (2009) Tree Data*

Description

Hypothetical data from Shipley (2009).

Details

Simulated data from a hypothetical study beginning in 1970 in which 20 sites `site` are chosen differing in latitude `lat`. Five individual trees `tree` of a particular species are chosen within each site. Each tree is followed every second year `year` until 2006 or until it dies (thus, repeated measures). At each site, in each sample year, and for each living individual you measure the cumulative degree

days DD until bud break, the Julian date (day of year) Date of bud break, the increase in stem diameter per tree Growth, and a binary variable indicating survival (1) or death (0) Survival during the subsequent growing season

Author(s)

Jon Lefcheck

References

Shipley, Bill. "Confirmatory path analysis in a generalized multilevel context." Ecology 90.2 (2009): 363-368.

http://www.esapubs.org/archive/ecol/E090/028/Shipley.dat

---

shipley2013 *Shipley (2013) Simulated Data*

Description

Hypothetical data from Shipley (2013).

Details

Simulated data containing 250 observations involving 10 individuals in each of 25 species. Five variables (x1,x2, etc.) have been measured on each individual.

Author(s)

Jon Lefcheck

References

Shipley, Bill. "Confirmatory path analysis in a generalized multilevel context." Ecology 90.2 (2009): 363-368.

Index