Package 'semPlot'

October 14, 2022

Type Package

Title Path Diagrams and Visual Analysis of Various SEM Packages' Output

Version 1.1.6

Maintainer Sacha Epskamp <mail@sachaepskamp.com>

Depends R (>= 2.15.0)

Suggests MplusAutomation (>= 0.5-3)

Imports qgraph (>= 1.2.4), lavaan (>= 0.5-11), sem (>= 3.1-0), plyr, XML, igraph (>= 0.6-3), lisrelToR, rockchalk, colorspace, corpcor, methods, OpenMx

ByteCompile yes

Description Path diagrams and visual analysis of various SEM packages' output.

URL https://github.com/SachaEpskamp/semPlot

License GPL-2

LazyLoad yes

NeedsCompilation no

Author Sacha Epskamp [aut, cre], Simon Stuber [ctb], Jason Nak [ctb], Myrthe Veenman [ctb], Terrence D. Jorgensen [ctb] (<https://orcid.org/0000-0001-5111-6773>)

Repository CRAN

Date/Publication 2022-08-10 07:30:02 UTC

R topics documented:

`semPlot-package` *semPlot*

Description

Path diagrams and visual analysis of various SEM packages' output. Path diagrams including visualizations of the parameter estimates can be plotted with `semPaths` and visualizations of the implied and observed correlation structures can be plotted using `semCors`. Finally, SEM syntax can be generated using`semSyntax`.

For plotting the graphs the `qgraph` package is used.

Author(s)

Sacha Epskamp (mail@sachaepskamp.com)

Maintainer: Sacha Epskamp <mail@sachaepskamp.com>

References

github.com/SachaEpskamp/semPlot

`cvregsem` *Bridge between cv_regsem output and sempaths*

Description

The package regsem (Jacobucci, 2017) is designed for a specific type of SEM called regularized structural equation modelling (RegSEM). For more information about RegSEM and the implementation in R we refer to the manual written by Jacobucci (2017).This function creates a bridge between the regsem and semplot packages, making it possible to use output from the regsem() and cv_regsem() functions to create models in sempaths.

Usage

```
## S3 method for class 'cvregsem'
semPlotModel(object,model,...)
```

Arguments

object	The regsem output
model	The cfa output used as input for the cv_regsem function
...	Arguments sent to 'lisrelModel', not used in other methods.

Value

A 'semPlotModel' object.

Author(s)

Sacha Epskamp <mail@sachaepskamp.com> Jason Nak <jasonnak@hotmail.com> Myrthe Veenman <myrthe.veenman@hotmail.com>

References

Jacobucci, R. (2017). regsem: Regularized Structural Equation Modeling. arXiv preprint arXiv:1703.08489.

See Also

semPlotModel semPaths

Examples

```
## Example of fitting and plotting a cv_regsem model in semPaths

#library(psych)
#library(lavaan)
#library(regsem)

# use a subset of the BFI
#bfi2 <- bfi[1:250,c(1:5,18,22)]
#bfi2[,1] <- reverse.code(-1,bfi2[,1])

# specify a SEM model
#mod <- "
#f1 =~ NA*A1+A2+A3+A4+A5+O2+N3
#f1~~1*f1
#"

# fit the model
#fit <- cfa(mod, bfi2)
#out.reg <- cv_regsem(fit, type="lasso", pars_pen=c(1:7), n.lambda=23, jump =.05)

# plot the model
#semPaths(semPlotModel.cvregsemplot(object = out.reg, model = fit))
```

`Imin` *Helper function to substract matrix from identity matrix and take inverse.*

Description

This function can be used to more easilly compute I - X or (I - X)^(-1), which are common in SEM models.

Usage

```
Imin(x, inverse = FALSE)
```

Arguments

`x`	A matrix
`inverse`	Logical, should the inverse be taken?

Author(s)

Sacha Epskamp <mail@sachaepskamp.com>

`lisrelModel` *Construct SEM model using LISREL matrix specification.*

Description

This function creates a 'semPlotModel' object using matrices of the extended LISREL model (Joreskog & Sorbom, 1996). This function has two main purposes. First, it can be used to easilly create path diagrams of arbitrary SEM models without having to run an actual analysis. And second, it is specifically designed to work with the output of the 'lisrelToR' package (using `do.call(lisrelModel,output$matrices)`). Using `semPaths` or `semPlotModel` on the file path of a LISREL output file will automatically first run `readLisrel` and then this function.

Usage

```
lisrelModel(LY, PS, BE, TE, TY, AL, manNamesEndo, latNamesEndo, LX, PH, GA, TD,
  TX, KA, manNamesExo, latNamesExo, ObsCovs, ImpCovs, setExo, modelLabels = FALSE,
  reduce)
```

Arguments

LY	Specification of the Lambda-Y matrix. See details.
PS	Specification of the Psi matrix. See details.
BE	Specification of the Beta matrix. See details.
TE	Specification of the Theta-Epsilon matrix. See details.
TY	Specification of the Tau-Y matrix. See details.
AL	Specification of the Alpha matrix. See details.
manNamesEndo	Character vector of names for the endogenous manifests.
latNamesEndo	Character vector of names for the endogenous latents.
LX	Specification of the Lambda-X matrix. See details.
PH	Specification of the Phi matrix. See details.
GA	Specification of the Gamma matrix. See details.
TD	Specification of the Theta-Delta matrix. See details.
TX	Specification of the Tau-X matrix. See details.
KA	Kappa
manNamesExo	Character vector of names for the exogenous manifests.
latNamesExo	Character vector of names for the exogenous latents.
ObsCovs	The observed covariance matrix, or a list of such matrices for each group.
ImpCovs	The implied covariance matrix, or a list of such matrices for each group.
setExo	Logical. If TRUE the 'exogenous' variable in the Variables data frame is specified. This forces semPaths to not attempt to identify which variables are endogenous and exogenous.
modelLabels	Logical. If TRUE all labels are set to the LISREL model matrix terms, as expressions. When plotted with semPaths this requires the argument `as.expression=c("nodes","edges")`.
reduce	Logical indicating if the variable number should be reduced if multiple variables are named exactly the same. If TRUE (default) directed edges between nodes that are named the same are removed and the manifest node is kept, as this usually indicates a way to include manifest variables in regressions.

Details

The LISREL matrices can be assigned in various ways, depending on the amount of information that should be stored in the resulting model.

First, the a single matrix can be used. The values of this matrix correspond to the parameter estimates in the 'semPlotModel'. For multiple groups, a list of such matrices can be used.

to store more information, a named list of multiple matrices of the same dimensions can be used. Included in this list can be the following (but only estimates is nessesary):

`est` Parameter estimates

`std` standardized parameter estimates

`par` Parameter numbers. 0 indicating fixed variables and parameters with the same parameter number are constrained to be equal.

`fixed` Logical matrix indicating if the parameter is fixed.

If `std` is missing the function tries to compute standardized solutions (not yet working for intercepts). If `fixed` is missing it is computed from the `par` matrix. For multiple groups, a list containing such lists can be used.

The number of variables is extracted from the assigned matrices. Matrices that are not assigned are assumed to be empty matrices of the appropriate dimensions. e.g., Lambda-Y is assumed to be a 0 by 0 matrix if there are no endogenous variables.

Value

A 'semPlotModel' object.

Author(s)

Sacha Epskamp <mail@sachaepskamp.com>

References

Joreskog, K. G., & Sorbom, D. (1996). LISREL 8 user's reference guide. Scientific Software.

https://github.com/SachaEpskamp/lisrelToR

See Also

`semPlotModel` `semCors` `semPaths` `ramModel`

Examples

```
## Example of a Full LISREL model path diagram with the same number of exgenous
## and endogenous variables:

# Lambda matrices:
Loadings <- rbind(diag(1,2,2),diag(1,2,2),diag(1,2,2))

# Phi and Psi matrices:
LatVar <- diag(1,2,2)

# Beta matrix:
Beta <- matrix(0,2,2)
Beta[1,2] <- 1

# Theta matrices:
ManVar <- diag(1,nrow(Loadings),nrow(Loadings))

# Gamma matrix:
Gamma <- diag(1,2,2)

# Tau matrices:
ManInts <- rep(1,6)
```

```
# Alpha and Kappa matrices:
LatInts <- rep(1,2)

# Combine model:
mod <- lisrelModel(LY=Loadings,PS=LatVar,BE=Beta,TE=ManVar,
                   LX=Loadings,PH=LatVar,GA=Gamma,TD=ManVar,
                   TY=ManInts,TX=ManInts,AL=LatInts,KA=LatInts)

# Plot path diagram:
semPaths(mod, as.expression=c("nodes","edges"), sizeMan = 3, sizeInt = 1,
  sizeLat = 4)

# Plot path diagram with more graphical options:
semPaths(mod, as.expression=c("nodes","edges"), sizeMan = 3, sizeInt = 1,
  sizeLat = 4, label.prop=0.5, curve=0.5, bg="black", groups="latents",
    intercepts=FALSE, borders=FALSE, label.norm="O")
```

`modelMatrices` *Extract SEM model matrices*

Description

Create a `"semMatriModel"` object. Use `semMatrixAlgebra` to extract or compute with these models. The structure of `"semMatriModel"` objects is chosen such that they can be used to create a `semPlotModel-class` object using `do.call` in combination with `ramModel`, `lisrelModel` or `mplusModel` (not yet implemented). See details.

Usage

```
modelMatrices(object, model = "ram", endoOnly = FALSE)
```

Arguments

`object`	A `"semPlotModel"` object or any of the input types that can be used in `semPlotModel` directly.
`model`	Model to be used, `"mplus"`, `"ram"` or `"lisrel"`
`endoOnly`	Only needed when the model is `"lisrel"`, sets all variables to endogenous.

Details

The `"lisrel"` model uses the following matrix names: `LY`, `TE`, `PS`, `BE`, `LX`, `TD`, `PH`, `GA`, `TY`, `TX`, `AL` and `KA`. Regressions on manifest variables will cause dummy latents to be included in the model.

The `"mplus"` model uses the following matrix names: `Lambda`, `Nu`, `Theta`, `Kappa`, `Alpha`, `Beta`, `Gamma` and `Psi`.

The `"ram"` model uses the following matrix names: `F`, `A` and `S`.

Value

a "semMatriModel" object

Author(s)

Sacha Epskamp <mail@sachaepskamp.com>

See Also

semPlotModel semPlotModel-class semMatrixAlgebra lisrelModel ramModel

Examples

```
## Mplus user guide SEM example:
outfile <- tempfile(fileext=".out")
tryres <- try({
  download.file("http://www.statmodel.com/usersguide/chap5/ex5.11.html",outfile)
})

if (!is(tryres,"try-error")){
  # Plot model:
  semPaths(outfile, intercepts = FALSE)

  # Extract RAM:
  RAM <- modelMatrices(outfile, "ram")
  semPaths(do.call(ramModel, RAM), as.expression = "edges", intercepts = FALSE)

  # Extract LISREL:
  LISREL <- modelMatrices(outfile, "lisrel")
  semPaths(do.call(lisrelModel, LISREL), as.expression = "edges", intercepts = FALSE)
}
```

---

ramModel *Construct SEM model using RAM matrix specification.*

Description

This function creates a 'semPlotModel' object using matrices of the RAM model (McArdle & McDonald, 1984).

Usage

```
ramModel(A, S, F, M, manNames, latNames, Names, ObsCovs, ImpCovs, modelLabels = FALSE)
```

Arguments

A	Specification of the assymmetric (A) matrix, see details.
S	Specification of the symmetric (S) matrix, see details.
F	Specification of the filter (F) matrix, see details.
M	Specification of the means (M) vector, see details.
manNames	Character vector of the manifest names.
latNames	Character vector of the latent names.
Names	Character vector containing all names. Defaults to c(manNames,latNames).
ObsCovs	Observed covariancem matrix.
ImpCovs	Implied covariancem matrix.
modelLabels	Logical. If TRUE all latents are named l1, l2, ... and all manifests m1, m2, ...

Details

The matrices can be assigned in various ways, depending on the amount of information that should be stored in the resulting model.

First, the a single matrix can be used. The values of this matrix correspond to the parameter estimates in the 'semPlotModel'. For multiple groups, a list of such matrices can be used.

to store more information, a named list of multiple matrices of the same dimensions can be used. Included in this list can be the following (but only estimates is nessesary):

`est` Parameter estimates

`std` standardized parameter estimates

`par` Parameter numbers. 0 indicating fixed variables and parameters with the same parameter number are constrained to be equal.

`fixed` Logical matrix indicating if the parameter is fixed.

If `std` is missing the function tries to compute standardized solutions (not yet working for intercepts). If `fixed` is missing it is computed from the `par` matrix. For multiple groups, a list containing such lists can be used.

The number of variables is extracted from the assigned matrices.

Value

A 'semPlotModel' object.

Author(s)

Sacha Epskamp <mail@sachaepskamp.com>

References

McArdle, J. J., & McDonald, R. P. (1984). Some algebraic properties of the reticular action model for moment structures. British Journal of Mathematical and Statistical Psychology, 37(2), 234-251.

See Also

semPlotModel semCors semPaths lisrelModel

---

regsem *Bridge between regsem output and sempaths*

Description

The package regsem (Jacobucci, 2017) is designed for a specific type of SEM called regularized structural equation modelling (RegSEM). For more information about RegSEM and the implementation in R we refer to the manual written by Jacobucci (2017).This function creates a bridge between the regsem and semplot packages, making it possible to use output from the regsem() and cv_regsem() functions to create models in sempaths.

Usage

```
## S3 method for class 'regsem'
semPlotModel(object,...)
```

Arguments

object	The regsem output
...	Arguments sent to 'lisrelModel', not used in other methods.

Value

A 'semPlotModel' object.

Author(s)

Sacha Epskamp <mail@sachaepskamp.com> Myrthe Veenman <myrthe.veenman@hotmail.com> Jason Nak <jasonnak@hotmail.com>

References

Jacobucci, R. (2017). regsem: Regularized Structural Equation Modeling. arXiv preprint arXiv:1703.08489.

See Also

semPlotModel semPaths

Examples

```
## Not run:
## Example of fitting and plotting a regsem model in semPaths
library(psych)
library(lavaan)
library(regsem)

# use a subset of the BFI
bfi2 <- bfi[1:250,c(1:5,18,22)]
bfi2[,1] <- reverse.code(-1,bfi2[,1])

# specify a SEM model
mod <- "
f1 =~ NA*A1+A2+A3+A4+A5+O2+N3
f1~~1*f1
"

# fit the model
fit <- cfa(mod, bfi2)
out.reg <- regsem(fit, type="lasso", pars_pen=c(1:7))

# plot the model
semPaths(semPlotModel.regsem(object = out.reg))

## End(Not run)
```

`semCors` *Visually inspect implied and observed correlations*

Description

This function is still in devellopment.

Usage

```
semCors(object, include, vertical = TRUE, titles = FALSE, layout, maximum, ...)
```

Arguments

`object`	A `semPlotModel` object
`include`	What to include? Can be `"observed"`, `"implied"` or `"difference"`, or a vector containing both. Defaults to showing observed and implied covariances.
`vertical`	Should the layout be vertical or horizontal?
`titles`	Logical, should titles indicating the group and observed/implied correlations be plotted?
`layout`	An optional layout matrix send to `qgraph`.

maximum	The maximum values as used in qgraph. Defaults to 1 for observed and implied covariances and 0.1 for difference graph. Important to note: Setting this lower than any of the covariances when comparing observed and implied correlations makes these graphs NOT interpretable.
...	Arguments sent to qgraph

Author(s)

Sacha Epskamp <mail@sachaepskamp.com>

`semMatrixAlgebra` *Extract or calculate with model matrices*

Description

This function can be used to extract or calculate with model matrices given a `"semMatriModel"` object (from `modelMatrices`) or a `"semPlotModel"` object or any of the input types that can be used in `semPlotModel` directly.

If the model is not specified it is attempted to be identified by the given algebra.

Usage

```
semMatrixAlgebra(object, algebra, group, simplify = TRUE, model, endoOnly = FALSE)
```

Arguments

object	A `"semMatriModel"` object (from `modelMatrices`) or a `"semPlotModel"` object or any of the input types that can be used in `semPlotModel` directly.
algebra	An R expression to use.
group	Groups the algebra should be used on. If more than one a list is returned with the result for each group.
simplify	If TRUE and only one group is used, return output as is instead of in a list.
model	Model to be used in `modelMatrices`, `"mplus"`, `"ram"` or `"lisrel"`
endoOnly	Only needed when the model is `"lisrel"`, sets all variables to endogenous.

Details

The `"lisrel"` model uses the following matrix names: `LY`, `TE`, `PS`, `BE`, `LX`, `TD`, `PH`, `GA`, `TY`, `TX`, `AL` and `KA`.

The `"mplus"` model uses the following matrix names: `Lambda`, `Nu`, `Theta`, `Kappa`, `Alpha`, `Beta`, `Gamma` and `Psi`.

The `"ram"` model uses the following matrix names: `F`, `A` and `S`.

Value

A list containing output per group

Author(s)

Sacha Epskamp <mail@sachaepskamp.com>

See Also

semPlotModel semPlotModel-class modelMatrices lisrelModel ramModel

Examples

```
## Mplus user guide SEM example:
outfile <- tempfile(fileext=".out")
tryres <- try({
  download.file("http://www.statmodel.com/usersguide/chap5/ex5.11.html",outfile)
})

if (!is(tryres,"try-error")){
# Plot model:
semPaths(outfile,intercepts=FALSE)

# Obtain latent regressions (mplus)
semMatrixAlgebra(outfile, Beta)

# mplus model implied covariance:
mat1 <- semMatrixAlgebra(outfile,
  Lambda %*% Imin(Beta, TRUE) %*% Psi %*% t(Imin(Beta, TRUE)) %*% t(Lambda) + Theta)

# Lisrel model implied covariance:
mat2 <- semMatrixAlgebra(outfile,
  LY %*% Imin(BE, TRUE) %*% PS %*% t(Imin(BE, TRUE)) %*% t(LY) + TE, endoOnly = TRUE)

# RAM model implied covariance:
mat3 <- semMatrixAlgebra(outfile,
                 F %*% Imin(A,TRUE) %*% S %*% t(Imin(A, TRUE)) %*% t(F))

## Not run:
# Plot:
library("qgraph")

pdf("Models.pdf",width=15,height=5)
layout(t(1:3))
qgraph(round(cov2cor(mat1),5), maximum=1, edge.labels=TRUE, layout = "spring",
  cut = 0.4, minimum = 0.1)
title("Mplus model")
qgraph(round(cov2cor(mat2),5), maximum=1, edge.labels=TRUE, layout = "spring",
  cut = 0.4, minimum = 0.1)
title("LISREL model")
qgraph(round(cov2cor(mat3),5), maximum=1, edge.labels=TRUE, layout = "spring",
  cut = 0.4, minimum = 0.1)
title("RAM model")
dev.off()

## End(Not run)
```

```
# They are the same.
}
```

---

semPaths *Plot path diagram for SEM models.*

Description

This function creates a path diagram of a SEM model (or general linear model), which is then plotted using qgraph. Currently many different SEM programs and packages are supported. Please see my website (www.sachaepskamp.com) for more details on which packages are supported and what is supported for each package.

Usage

```
semPaths(object, what = "paths", whatLabels, style, layout = "tree",
  intercepts = TRUE, residuals = TRUE, thresholds = TRUE, intStyle = "multi",
  rotation = 1, curve, curvature = 1, nCharNodes = 3, nCharEdges = 3, sizeMan = 5,
   sizeLat = 8,  sizeInt = 2, sizeMan2, sizeLat2, sizeInt2, shapeMan, shapeLat,
  shapeInt = "triangle", ask, mar, title, title.color = "black", title.adj = 0.1,
  title.line = -1, title.cex = 0.8, include, combineGroups = FALSE, manifests,
  latents, groups, color, residScale, gui = FALSE, allVars = FALSE, edge.color,
  reorder = TRUE, structural = FALSE, ThreshAtSide = FALSE, thresholdColor,
  thresholdSize = 0.5, fixedStyle = 2, freeStyle = 1,
  as.expression = character(0), optimizeLatRes = FALSE, inheritColor = TRUE,
  levels, nodeLabels, edgeLabels, pastel = FALSE, rainbowStart = 0, intAtSide,
  springLevels = FALSE, nDigits = 2, exoVar, exoCov = TRUE, centerLevels = TRUE,
  panelGroups = FALSE, layoutSplit = FALSE, measurementLayout = "tree", subScale,
  subScale2, subRes = 4, subLinks, modelOpts = list(mplusStd = "std"),
  curveAdjacent = '<->', edge.label.cex = 0.6,  cardinal = "none",
 equalizeManifests = FALSE, covAtResiduals = TRUE, bifactor, optimPoints = 1:8 * (pi/4),
  ...)
```

Arguments

object	A "semPlotModel" object or any of the input types that can be used in semPlotModel directly.
what	What should the edges indicate in the path diagram? This function uses grepl to allow fuzzy matching and is not case sensitive. E.g., par will also match Parameters.
	path, diagram **or** mod This will display the model as an unweighted network (gray edges by default).
	est **or** par This will display the parameter estimates as weighted edges.
	stand **or** std This will display the standardized parameter estimates, if available, as weighted edges.

`eq` **or** `cons` This is the same graph as `path`. except that parameters with equality constraints are now colored. Parameters with the same color are constrained to be equal.

`col` This will create an unweighted graph of the path diagram, where edges are colored with a mix of the colors of connected nodes.

`whatLabels` What should the edge labels indicate in the path diagram? This function uses `grepl` to allow fuzzy matching and is not case sensitive. E.g., `par` will also match `Parameters`. Default depends on the `what` argument, defaulting to the respective elements in the list below for values of `what` in the list above.

`name`**,** `label`**,** `path` **or** `diagram` This will display the edge names as labels.

`est` **or** `par` This will display the parameter estimate in edge labels.

`stand` **or** `std` This will display the standardized parameter estimate in edge labels.

`eq` **or** `cons` This will display the parameter number in edge labels. 0 indicates the parameter is fixed, parameters with the same parameter number are constrained to be equal.

`no`**,** `omit`**,** `hide` **or** `invisible` Hides edge labels.

`style` The style to use. Currently only indicates what the (residual) variances look like. Use `"ram"`, `"mx"` or `"OpenMx"` for double headed selfloops and `"lisrel"` for single headed edges with no node as origin. Defaults to `"ram"` unless the input is a lisrel model.

`layout` A string indicating how the nodes should be placed. Similar to the 'layout' argument in `qgraph`. Can be one of the following strings.

tree The integrated tree-like layout. Places exogenous variables at the top and endogenous variables at the bottom. See 'details' for more details.

circle The same layout as "tree", except that afterwards the horizontal levels of the layout are placed in circles. Especially useful for models with a large number of manifest variables and a relatively small number of latent variables.

spring Calls the "spring" layout in `qgraph`, which uses the Fruchterman-reingold algorithm (Fruchterman & Reingold, 1991).

tree2 Calls the `layout.reingold.tilford` function from the igraph package (Csardi & Nepusz, 2006), which uses the Reingold-Tilford algorithm (Reingold & Tilford, 1981). Before calling the algorithm roots are chosen and a slightly modified version of the graph is used to produce consistent results. See 'details'.

circle2 The same layout as "tree2", except that afterwards the horizontal levels of the layout are placed in circles.

Other options If the assigned value is not in this list it is sent to `qgraph`. This allows for manual specification of the layout as well as using functions found in the 'igraph; library.

`intercepts` Logical, should intercepts be included in the path diagram?

`residuals` Logical, should residuals (and variances) be included in the path diagram?

`thresholds` Logical, should thresholds be included in the path diagram?

intStyle	Style of the intercepts. "multi" plots a separate unit vector node for each intercept and "single" plots a single unit vector node. Currently, "single" is not well supported and might lead to unexpected results.
rotation	An integer indicating the rotation of the layout when "tree" or "tree2" layout is used. 1, 2, 3 and 4 indicate that exogenous variables are placed at the top, left side, bottom and right side respectively.
curve	The curvature of the edges. In tree layouts this argument only curves the edges that are between nodes on the same level. e.g., correlations between exogenous manifest variables.
curvature	Sets the strength of scaling in curvature for curved edges at the same horizontal level in tree layouts. The curve will be set to curve + curvature * n / max(n), where n is the number of nodes in between the two connected nodes.
nCharNodes	Number of characters to abbreviate node labels to (using abbreviate). Set to 0 to omit abbreviation.
nCharEdges	Number of characters to abbreviate edge labels to (using abbreviate). Set to 0 to omit abbreviation.
sizeMan	Width of the manifest nodes, sent to the 'vsize' argument in qgraph.
sizeLat	Width of the latent nodes, sent to the 'vsize' argument in qgraph.
sizeInt	Width of the unit vector nodes, sent to the 'vsize' argument in qgraph.
sizeMan2	Height of the manifest nodes, sent to the 'vsize2' argument in qgraph.
sizeLat2	Height of the latent nodes, sent to the 'vsize2' argument in qgraph.
sizeInt2	Height of the unit vector nodes, sent to the 'vsize2' argument in qgraph.
shapeMan	Shape of the manifest nodes, sent to the 'shape' argument in qgraph. Defaults to "square" or "rectangle" if width and height differ.
shapeLat	Shape of the latent nodes, sent to the 'shape' argument in qgraph. Defaults to "circle" or "ellipse" if width and height differ.
shapeInt	Shape of the constant nodes, sent to the 'shape' argument in qgraph. Defaults to "triangle".
ask	Specifies the 'ask' parameter in par. Defaults to TRUE if multiple groups are in the model.
mar	Same as the 'mar' argument in qgraph. By default this argument is based on the values of 'rotation', 'style' and 'title'.
title	Logical, should titles be plotted of the group names above each plot?
title.color	Color of the titles.
title.adj	Adjustment of title as used by 'adj' in par.
title.line	Line of title as used by 'line' in title.
title.cex	Size of title as used by 'cex.main' in par.
include	Integer vector indicating which groups should be included in the output. e.g., to only plot a diagram for the first group use include = 1.
combineGroups	Logical. If TRUE all groups are combined in the same path diagram.

`manifests` A character vector in which every element is the name of a manifest variable in the model. This argument can be used to overwrite the order in which nodes are plotted in the graph if `reorder = FALSE`

`latents` A character vector in which every element is the name of a latent variable in the model. This argument can be used to overwrite the order in which nodes are plotted in the graph if `reorder = FALSE`

`groups` Groups nodes that should be colored the same, similar to the 'groups' argument in `qgraph` with a few exceptions. Should be a list containing in each element the names (instead of numbers as in qgraph) of nodes that belong together. Nodes that are indicated to belong to a group will be assigned the same color, as given by the 'color' argument. Nodes not belonging to a group will be assigned the color "", which indicates that they will inherit a mix of the colors of connected nodes (or white, if no connected nodes are colored.)

In addition, this argument can be assigned a single character: "manifests", "latents" or "both" to make a single group for each manifest, latent or both manifest and latent variables. e.g., `groups = "latents"` will color each latent variable uniquely, and color all manifest variables a mixture of the colors of latents they load on.

`color` Controls the color of nodes. Similar to 'color' in `qgraph`. A color vector indicating the color for each group, a single color character indicating the color for all nodes or a color vector indicating the color for each node separately. Can also be a list contaning one or more of the following elements (using fuzzy matching):

man The colors for manifest nodes

lat The colors for latent nodes

int The color for intercepts

`residScale` The size of residual edges if `style = "lisrel"`. Defaults to two times the value of 'sizeMan'.

`gui` Not yet implemented.

`allVars` Logical. If TRUE all variables are plotted in the path diagrams for each group. If FALSE only variables are plotted that are used in the group.

`edge.color` A value indicating the color of all edges or a vector indicating the color of each edge. Useful for manually overwriting edge colors.

`reorder` Logical. Should manifest variables be reordered to be near latent factors they are most connected to in the "tree" layout? If FALSE manifest variables are placed in the order they appear in the Pars.

`structural` Logical. Set this to TRUE to only show the structural model (omit all manifest variables.)

`ThreshAtSide` Logical. If TRUE, thresholds are plotted as small lines at the side of manifest nodes, otherwise they are plotted as lines inside the nodes.

`thresholdColor` Color of the threshold lines. Defaults to "black"

`thresholdSize` Size of threshold bars relative to the size of the node.

`fixedStyle` A vector of length one or two specifying the color and line type (same as 'lty' in `par`) of fixed parameters. Can be both character and numeric. If one of the

elements encodes a color it is used to overwrite the color of fixed edges, and if an element can be coerced to a numeric it is used to encode the line type.

For example, `fixedStyle = c("red",3)` specifies that all fixed parameters should be visualized with a red edge with `lty=3`

`freeStyle` Same as 'fixedStyle' but for free parameters instead.

`as.expression` A character vector indicating which labels should be treated as an `expression`, so that mathematical notation and Greek letters can be used in the path diagram. If this vector contains `"nodes"` all node labels are converted to expressions, and if this vector contains `"edges"` all node labels are converted to expressions. Defaults to `"edges"` only if the input is a Lisrel model.

`optimizeLatRes` Logical. If this is TRUE, the angle of the incoming residuals on latent variables is attempted to be optimally chosen so its position conflicts with the least amount of connected edges.

`inheritColor` Logical, should uncolored nodes obtain a mix of connected colored nodes? Defaults to `TRUE`.

`levels` A numeric vector usually of length 4. Controls the relative vertical position of variable levels (exogenous and endogenous latents and manifests) under default rotation in tree and circle layouts. This can be used to control the spacing between these levels. e.g., `c(1,5,6,7)` will create more space between endogenous manifests and latents.

`nodeLabels` A vector or list to manually overwrite the node labels. Can include expressions.

`edgeLabels` A vector or list to manually overwrite the edge labels. Can include expressions.

`pastel` Logical, should default colors (for groups or edge equality constraints) be chosen from pastel colors? If TRUE then `rainbow_hcl` is used.

`rainbowStart` A number between 0 and 1 indicating the offset used in rainbow functions for default node coloring.

`exoVar` Should variances of truely exogenous variables (no incomming directed edge) be plotted? Defaults to `TRUE` unless `style = "lisrel"`.

`intAtSide` Logical to control if intercepts should be plotted to the side of manifest nodes or at the bottom/top. Defaults only to FALSE if 'residuals=FALSE'.

`springLevels` Logical indicating if the placement on horizontal levels with `tree3` layout should be determined by a force embedded algorithm.

`nDigits` Number of digits to round numeric values to.

`exoCov` Should covariances between truely exogenous variables (no incomming directed edge) be plotted? Defaults to `TRUE`.

`centerLevels` Only used if `layout` is set to `"tree2"`, should each level be centered? Defaults to `TRUE`

`panelGroups` Logical to automatically create a panel plot of multiple group models. Defaults to FALSE.

`layoutSplit` Logical that can be used to split computing of layout between structural and measurment models. This is very useful in more complicated models where the structural part is best shown by using a spring layout.

`measurementLayout`
: Logical indicating the layout algorithm to use for measurement models if `layoutSplit = TRUE` (the structural model will obtain a layout given by the `layout` argument).

`subScale`
: Width of submodels (measurment models) if `layoutSplit = TRUE`.

`subScale2`
: Height of submodels (measurment models) if `layoutSplit = TRUE`.

`subRes`
: Integer indicating the resolution of which measurment models can be rotated around their corresponding latent variable. The default, 4, indicates that they can be placed only to polar coordinates. Set to 360 to allow every angle of rotation.

`subLinks`
: Vector of variables to link to. Currently not well supported so avoid using this argument.

`modelOpts`
: A lists containing arguments sent to `semPlotModel` in case the input is not of class `semPlotModel`.

`curveAdjacent`
: What edges between adjacent horizontal nodes be curved? Can be `'<->'` or `'cov'` to indicate bidirectional covariances, `'->'` or `'reg'` for directed regressions or a vector containing both.

`edge.label.cex`
: Controls the font size of the edge labels. Same as in `qgraph` except that the default is now 0.8.

`cardinal`
: Should edges in a tree layout connect to the four cardinal points of one of the borders of the node rather than point to the center of the node? Can be set to `TRUE` or `"all"` to enamble this behavior for all edges and `FALSE` or `"none"` to disable this behavior for all edges. Alternatively a vector with strings can be specified in which each string specifies a certain group of edges. Fuzzy matching is used on the strings `"exo"` for edges with the first node being exogenous (or indicator of exogenous latent), `endo` for edges with first node being endogeonous, `manifest` for edges connected to any manifest node, `latent` for edges connected to any latent node, `cov` for covariances, `reg` for regressions, `load` for factor-loadings, `source` for only the start of an edge and `end` for only the end of a node. These strings can be combined at will. For example, `cardinal = c("exo cov","load end")` (the default) or equivelantly `cardinal = c("exogenous covariances","source of loadings")` will only cardenalize the edges that represent exogenous covariances or the end of factor loadings.

`equalizeManifests`
: Logical. Should the distances between manifest nodes in the `tree1` layout be equalized? Defaults to `TRUE`

`covAtResiduals`
: Logical, should covariances be drawn at the start of residuals when `style="lisrel"` is used? Defaults to TRUE.

`bifactor`
: A string vector containing the name(s) of the general bifactor(s). This will automatically create a bifactor plot.

`optimPoints`
: A vector of radians residuals can optimize to if `optimizeLatRes = TRUE`

`...`
: Arguments sent to the `qgraph` function. These arguments can further control the output of the graph. Some usefull arguments in drawing path diagrams are:

  edge.width Scales the edge width and arrow size of the plot. These can also be manually set using 'esize' and 'asize'.

node.width Scales the width of nodes and also the height if shapes circle and square are used. Can also be a vector with scalar for each node.

node.height Scales the height of nodes. Can also be a vector with scalar for each node. Not used with circle and square shapes.

esize Size of the largest edge (or what it would be if there was an edge with weight maximum). Defaults to: max((-1/72)*(nNodes)+5.35,1) for weighted graphs and 2 for unweighted graphs. In directed graphs these values are halved.

asize Size of the arrowhead. Defaults to 2 for graphs with more than 10 nodes and 2 to smaller graphs.

minimum Edges with absolute weights under this value are omitted. Defaults to 0 for graphs with less than 50 nodes or 0.1 for larger graphs.

maximum qgraph regards the highest of the maximum or highest absolute edge weight as the highest weight to scale the edge widths too. To compare several graphs, set this argument to a higher value than any edge weight in the graphs (typically 1 for correlations).

cut In weighted graphs, this argument can be used to cut the scaling of edges in width and color saturation. Edges with absolute weights over this value will have the strongest color intensity and become wider the stronger they are, and edges with absolute weights under this value will have the smallest width and become vaguer the weaker the weight. If this is set to NULL, no cutoff is used and all edges vary in width and color. Defaults to NULL for graphs with less then 50 nodes and 0.3 to larger graphs.

details Logical indicating if minimum, maximum and cutoff score should be printed under the graph. Defaults to FALSE.

mar A vector of the form c(bottom, left, top, right) which gives the margins. Works similar to the argument in par(). Defaults to c(3,3,3,3)

filetype A character containing the file type to save the output in. "R" outputs in a new R window, "pdf" creates a pdf file. "svg" creates a svg file (requires RSVGTipsDevice). "tex" creates LaTeX code for the graph (requires tikzDevice). 'jpg', 'tiff' and 'png' can also be used. If this is given any other string (e.g. filetype="") no device is opened. Defaults to 'R' if the current device is the NULL-device or no new device if there already is an open device. A function such as x11 can also be used

filename Name of the file without extension

width Width of the plot, in inches

height Height of the plot, in inches

normalize Logical, should the plot be normalized to the plot size. If TRUE (default) border width, vertex size, edge width and arrow sizes are adjusted to look the same for all sizes of the plot, corresponding to what they would look in a 7 by 7 inches plot if normalize is FALSE.

DoNotPlot Runs qgraph but does not plot. Useful for saving the output (i.e. layout) without plotting

plot Logical. Should a new plot be made? Defaults to TRUE. Set to FALSE to add the graph to the existing plot.

rescale Logical. Defines if the layout should be rescaled to fit the -1 to 1 x and y area. Defaults to TRUE. Can best be used in combination with plot=FALSE.

label.cex Scalar on the label size.

label.color Character containing the color of the labels, defaults to "black"

borders Logical indicating if borders should be plotted, defaults to TRUE.

border.color Color vector indicating colors of the borders. Is repeated if length is equal to 1. Defaults to "black"

border.width Controls the width of the border. Defaults to 2 and is comparable to 'lwd' argument in 'points'.

polygonList A list contaning named lists for each element to include polygons to lookup in the `shape` arguments. Each element must be named as they are used in `shape` and contain a list with elements `x` and `y` contaning the coordinates of the polygon. By default `ellipse` and `heart` are added to this list. These polygons are scaled according to `vsize` and `vsize2`

vTrans Transparency of the nodes, must be an integer between 0 and 255, 255 indicating no transparency. Defaults to 255

label.prop Controls the proportion of the width of the node that the label rescales to. Defaults to 0. 9.

label.norm A single string that is used to normalize label size. If the width of the label is lower than the width of the hypothetical label given by this argument the width of label given by this argument is used instead. Defaults to "OOO" so that every label up to three characters has the same fontsize.

label.scale Logical indicating if labels should be scaled to fit the node. Defaults to TRUE.

label.font Integer specifying the label font of nodes. Can be a vector with value for each node

posCol Color of positive edges. Can be a vector of two to indicate color of edges under 'cut' value and color of edges over 'cut' value. If 'fade is set to TRUE the first color will be faded the weaker the edge weight is. If this is only one element this color will also be used for edges stronger than the 'cut' value. Defaults to c("#009900","darkgreen")

negCol Color of negative edges. Can be a vector of two to indicate color of edges under 'cut' value and color of edges over 'cut' value. If 'fade is set to TRUE the first color will be faded the weaker the edge weight is. If this is only one element this color will also be used for edges stronger than the 'cut' value. Defaults to c("#BF0000","red")

unCol Color to indicate the default edge color of unweighted graphs. Defaults to "#808080".

colFactor Exponent of transformation in color intensity of relative strength. Defaults to 1 for linear behavior.

trans In weighted graphs: logical indicating if the edges should fade to white (FALSE) or become more transparent (TRUE; use this only if you use a background). In directed graphs this is a value between 0 and 1 indicating the level of transparency. (also used as 'transparency')

fade if TRUE (default) and if 'edge.color' is assigned, transparency will be added to edges that are not transparent (or for which no transparency has been assigned) relative to the edge strength, similar if 'trans' is set to TRUE.

loop This can be used to scale the size of the loop. defaults to 1.

curvePivot Quantile to pivot curves on. This can be used to, rather than round edges, make straight edges as curves with "knicks" in them. Can be logical or numeric. `FALSE` (default) indicates no pivoting in the curved edges, a number indicates the quantile (and one minus this value as quantile) on which to pivot curved edges and `TRUE` indicates a value of 0.1.

curvePivotShape The shape of the curve around the pivots, as used in `xspline`. Defaults to `0.25`.

edge.label.bg Either a logical or character vector/matrix. Indicates the background behind edge labels. If TRUE (default) a white background is plotted behind each edge label. If FALSE no background is plotted behind edge labels. Can also be a single color character, a vector or matrix of color vectors for each edge.

edge.label.position Vetor of numbers between 0 and 1 controlling the relative position of each edge label. Defaults to 0.5 for placing edge labels at the middle of the edge.

edge.label.font Integer specifying the label font of edges. Can be a vector or matrix with value for each node

layout.par A list of arguments passed to `qgraph.layout.fruchtermanreingold` when layout="spring" or to an igraph function when such a function is assigned to 'layout'

bg If this is TRUE, a background is plotted in which node colors cast a light of that color on a black background. Can also be a character containing the color of the background Defaults to FALSE

bgcontrol The higher this is, the less light each node gives if bg=TRUE. Defaults to 6.

bgres square root of the number of pixels used in bg=TRUE, defaults to 100.

pty See 'par'

font Integer specifying the default font for node and edge labels

arrows A logical indicating if arrows should be drawn, or a number indicating how much arrows should be drawn on each edge. If this is TRUE, a simple arrow is plotted, if this is a number, arrows are put in the middle of the edges.

arrowAngle Angle of the arrowhead, in radians. Defaults to pi/8 for unweighted graphs and pi/4 for weighted graphs.

asize Size of the arrowhead. Defaults to 2 for graphs with more than 10 nodes and 2 to smaller graphs.

open Logical indicating if open (TRUE) or closed (FALSE) arrowheads should be drawn.

weighted Logical that can be used to force either a weighted graph (TRUE) or an unweighted graph(FALSE).

XKCD If set to TRUE the graph is plotted in XKCD style based on http://stackoverflow.com/a/12680841/

Details

The default "tree" layout under default rotation places the nodes in one of four horizontal levels. At the top the exogenous manifest variables, under that the exogenous latent variables, under that the endogenous latent variables and at the bottom the endogenous manifest variables. If one of these kinds of variables does not exist its level is omitted. Afterwards, the rotation argument will rotate the graph and the "circle" layout will make the layout circular using these levels as nested circles.

If not manually set (see semPlotModel-edit), semPath will automatically try to set the endogenous and exogenous variables, such that the resulting layout looks good. A latent variable is identified as *exogenous* if it is not on the right hand side of a directed edge (-> or ~>) with another latent variable as node of origin. A manifest variable is set as *exogenous* if it is only connected, in any way, to exogenous latent variables and if it is not the right hand side (dependent variable) of a regression edge (~>). If all variables are set to exogenous this way, they are all set to endogenous for consistency in the layouts. Afterwards, manifest variables only used in formative measurement models (only outgoing directed edges to latents) are set to exogenous again so that MIMIC models are displayed properly.

Intercepts are placed on the same level as the variable, either on the left or right side of the node (pointing outward from the center). Residuals for manifest variables are placed at the top or bottom (for exogenous and endogenous manifests respectively). Residuals of latents are placed at the bottom or top respectively for exogenous and endogenous variables, but is switched if the latent is not connected to a manifest. Residuals for the leftmost and rightmost latent are placed at the left and right side respectively, or diagonal if the latent is connected to an intercept.

The "tree2" and "circle2" layouts call the layout.reingold.tilford function from the igraph package. As roots are used the first available variables of the following list:

- Intercepts of exogenous manifests
- Exogenous manifest
- Intercepts of exogenous latents
- Exogenous latents
- Interceots of endogenous latents
- Endogenous latents
- Intercepts of endogenous manifests
- The endogenous manifest with the most outgoing edges (this should not be possible by default, but can be manually set)
- The most connected endogenous manigest.

To compute an optimal layout layout.reingold.tilford is run on a slightly altered version of the path diagram. In this version, the direction of edges from all intercepts that are not roots is reversed, the direction of all edges leading to exogenous manifests is reversed and all bidirectional edges are removed.

Value

A "qgraph" object as returned by qgraph. This object can be used to alter the graph (such as manually redefining the layout) and to plot the graph again with different arguments.

If there are multiple groups a list is returned with a "qgraph" object for each path diagram that has been produced.

Author(s)

Sacha Epskamp <mail@sachaepskamp.com>

References

Fruchterman, T. & Reingold, E. (1991). Graph drawing by force-directed placement. Software - Pract. Exp. 21, 1129-1164.

Reingold, E and Tilford, J (1981). Tidier drawing of trees. IEEE Trans. on Softw. Eng., SE-7(2):223-228.

Csardi G, Nepusz T (2006). The igraph software package for complex network research, InterJournal, Complex Systems 1695. http://igraph.sf.net

See Also

qgraph semPlotModel semPlotModel-class semCors lisrelModel semSyntax

Examples

```
 # Regression analysis with interaction effects ----------------------------

# A silly dataset:
X <- rnorm(100)
Y <- rnorm(100)
Z <- rnorm(1)*X + rnorm(1)*Y + rnorm(1)*X*Y
DF <- data.frame(X,Y,Z)

# Regression including interaction:
res <- lm(Z ~ X*Y, data = DF)

# Path diagram:
semPaths(res, intAtSide=TRUE)

# Standardized estimates:
semPaths(res,"std","hide", intAtSide=TRUE)

# Simple CFA ----------------------------------------
library("lavaan")
example(cfa)

semPaths(fit, 'std', 'est', curveAdjacent = TRUE, style = "lisrel")


# MIMIC model ---------------------------------------------------
## Lavaan
## Not run:
library("lavaan")

# Example 5.8 from mplus user guide:
Data <- read.table("http://www.statmodel.com/usersguide/chap5/ex5.8.dat")
names(Data) <- c(paste("y", 1:6, sep=""),
                 paste("x", 1:3, sep=""))
```

```
# Model:
model.Lavaan <- 'f1 =~ y1 + y2 + y3
f2 =~ y4 + y5 + y6
f1 + f2 ~ x1 + x2 + x3 '

# Run Lavaan:
library("lavaan")
fit <- lavaan:::cfa(model.Lavaan, data=Data, std.lv=TRUE)

# Plot path diagram:
semPaths(fit,title=FALSE)

# Omit exogenous covariances:
semPaths(fit,title=FALSE, exoVar = FALSE, exoCov = FALSE)

# Standardized parameters:
semPaths(fit,"std", edge.label.cex = 0.5, exoVar = FALSE,
  exoCov = FALSE)

## Mplus

# Same model, now using mplus output:
outfile <- tempfile(fileext=".out")
download.file("http://www.statmodel.com/usersguide/chap5/ex5.8.html",outfile)

# Plot model:
semPaths(outfile,intercepts=FALSE)
# Note that mplus did not report the fixed variances of the exogenous variables.




# Thresholds  ----------------------------------------------------
## Lavaan

# Example 5.8 from mplus user guide:
Data <- read.table("http://www.statmodel.com/usersguide/chap5/ex5.2.dat")
names(Data) <- c("u1","u2","u3","u4","u5","u6")
Data <- as.data.frame(lapply(Data, ordered))

# Lavaan model:
model <- ' f1 =~ u1 + u2 + u3; f2 =~ u4 + u5 + u6 '

# Run Lavaan:
fit <- lavaan::cfa(model, data=Data)

# Plot path diagram:
semPaths(fit,intercepts=FALSE)

## Mplus

# Same model, now using mplus output:
```

```
outfile <- tempfile(fileext=".out")
download.file("http://www.statmodel.com/usersguide/chap5/ex5.2.html",outfile)

# Plot model:
semPaths(outfile)




# OpenMx ----------------------------------------------------------
# To install OpenMx see:
# http://openmx.psyc.virginia.edu/

library("OpenMx")

# Example from mxRun help page:
# Create and run the 1-factor CFA on the openmx.psyc.virginia.edu front page
data(demoOneFactor)  # load the demoOneFactor dataframe
manifests <- names(demoOneFactor) # set the manifest to the 5 demo variables
latents   <- c("G")  # define 1 latent variable
model <- mxModel("One Factor", type="RAM",
  manifestVars = manifests,
latentVars   = latents,
mxPath(from=latents  , to=manifests),
mxPath(from=manifests, arrows=2),
mxPath(from=latents  , arrows=2, free=FALSE, values=1.0),
mxData(cov(demoOneFactor), type="cov", numObs=500)
)
model <- mxRun(model) #run model, returning the result

# Plot with colors from OpenMx front page:
semPaths(model, color = list(
              lat = rgb(245, 253, 118, maxColorValue = 255),
              man = rgb(155, 253, 175, maxColorValue = 255)),
         mar = c(10, 5, 10, 5))

## Factor Analysis:
source("http://openmx.ssri.psu.edu/docs/OpenMx/latest/_static/demo/TwoFactorModel_PathCov.R")
semPaths(twoFactorFit, layout = "tree2")

# Multi-group analysis ------------------------------------------
## LISREL:
# Download measurment invariance example:
modFile <- tempfile(fileext=".OUT")
download.file("http://sachaepskamp.com/files/mi1.OUT",modFile)
layout(t(1:2))
semPaths(modFile,"eq",ask=FALSE, intAtSide = TRUE, mar = c(8, 1, 5, 1))
# Color indicates equality constraints.

## End(Not run)
```

semPlot-tricks *Tricks that can be used in semPlot.*

Description

Use a list contaning several SEM objects (from any source) to plot them as the same model. Also, the '+' operator can be used to combine two models, including in calls in semPaths and semPlotModel. See examples.

Usage

```
## S3 method for class 'semPlotModel'
x + y
## S3 method for class 'list'
semPlotModel(object, ...)
```

Arguments

x	A "semPlotModel" object
y	A "semPlotModel" object
object	An object contaning the result of a SEM or GLM analysis, or a string contaning the file path to the output file of a sEM program.
...	Not used.

Author(s)

Sacha Epskamp <mail@sachaepskamp.com>

See Also

semPlotModel semPaths semCors

Examples

```
# A silly dataset:
A <- rnorm(100)
B <- A + rnorm(100)
C <- B + rnorm(100)
DF <- data.frame(A,B,C)

# Two regressions:
res1 <- lm(B ~ C, data = DF)
res2 <- lm(A ~ B + C, data = DF)

# Plot both in the same path diagram in two ways:
semPaths(res1 + res2, "model", "est", intercepts=FALSE)
semPaths(list(res1,res2), "model", "est", intercepts=FALSE)
```

`semPlotModel` *SEM model representation*

Description

Methods to read a SEM object and return a `semPlotModel-class` object.

Usage

```
## Default S3 method:
semPlotModel(object, ...)
## S3 method for class 'lm'
semPlotModel(object, ...)
## S3 method for class 'principal'
semPlotModel(object, ...)
## S3 method for class 'princomp'
semPlotModel(object, ...)
## S3 method for class 'loadings'
semPlotModel(object, ...)
## S3 method for class 'factanal'
semPlotModel(object, ...)

## S3 method for class 'lisrel'
semPlotModel(object, ...)

## S3 method for class 'mplus.model'
semPlotModel(object, mplusStd = c("std", "stdy", "stdyx"), ...)
## S3 method for class 'sem'
semPlotModel(object, ...)
## S3 method for class 'msem'
semPlotModel(object, ...)
## S3 method for class 'msemObjectiveML'
semPlotModel(object, ...)
semPlotModel_Amos(object)
semPlotModel_Onyx(object)
semPlotModel_lavaanModel(object, ...)
```

Arguments

`object`	An object contaning the result of a SEM or GLM analysis, or a string contaning the file path to the output file of a SEM program. Or a Lavaan model.
`mplusStd`	What standardization to use in Mplus models?
`model`	The original sem model (used in cvregsem)
`...`	Arguments sent to 'lisrelModel', not used in other methods.

Details

A detailed overview of which packages are supported and what is supported for each of them will soon be on my website.

Value

A "semPlotModel" object. See link{semPlotModel-class}

Author(s)

Sacha Epskamp <mail@sachaepskamp.com>

See Also

semPaths semCors semPlotModel-class

---

semPlotModel-class *Class* "semPlotModel"

Description

Representation of SEM models, can be used by semPaths, semCors and semSyntax. See semPlotModel-edit for utility functions on how to edit this model.

Objects from the Class

Objects can be created by calls of the form new("semPlotModel", ...).

Slots

Pars: Object of class "data.frame" indicating the parameters used in the SEM model. this must contain the following elements, in order:

label The name of the parameter, used as edge label in the graph.

lhs Name of the variable on the left hand side of the path.

edge String as indicator of the edge. This can be one of the following:

-> Factor loading

~> Regression. The same as '->' in that it results in a directed edge from the left hand side to the right hand side, but '~>' differs in that if the right hand side is manifest and the left hand side is an exogenous latent the right hand side is interpreted as an endogenous variable rather than an exogenous variable.

<-> (co)variance

int intercept, The left hand side should be "" and the right hand side indicates the variable to which the intercept belongs.

-- Undirected edge. Only used as dummy encoding and in cases the parameter can not be interpreted (usually this indicates something that is not yet supported)

rhs Name of the variable on the left hand side of the path.

`est` Parameter estimate.

`est` Standardized parameter estimate.

`group` Character of the name of the group the parameter belongs to.

`fixed` Logical indicating if the parameter is fixed.

`par` Parameter number. 0 indicates the parameter is fixed and parameters with the same parameter number are constrained to be equal.

`knot` Knot number. 0 indicates the edge is not knotted and edges with the same knot number are knotted together. Only used to indicate interactions in 'lm' models and can be omitted.

`Vars`**:** Object of class `"data.frame"` indicating the variables used in the SEM model. Must have the following elements:

name Name of the variable

manifest Logical indicating if the variable is manifest

exogenous Logical indicating if the variable is exogenous. If `NA`(the default) `semPaths` will attempt to detect which variables are exogenous.

`Thresholds`**:** Object of class `"data.frame"` indicating the thresholds in the SEM model. It is the same as `Pars` except it does not have the elements `'edge'` and `'rhs'`.

`Computed`**:** Object of class `"logical"` indicating if the SEM model was computed or if the object only indicates a structure.

`ObsCovs`**:** Object of class `"list"` containing observed covariance matrices for each group. If available.

`ImpCovs`**:** Object of class `"list"` containing implied covariance matrices for each group. If available.

`Original`**:** Object of class `"list"` containing the original object used as input (or multiple objects if the `'+'` operator was used to combine objects.)

Methods

No methods defined with class "semPlotModel" in the signature.

Author(s)

Sacha Epskamp <mail@sachaepskamp.com>

See Also

`semPlotModel` `semPaths` `semCors` `semSyntax` `semPlotModel-edit`

Examples

```
showClass("semPlotModel")
```

semPlotModel-edit *Functions to facilitate editting 'semPlotModel' objects.*

Description

These functions can be used to easilly call and edit parts of a semPlotModel-class object. Currently only manifest/latent and endgenous/exogenous node properties can be set.

Usage

```
exo(x)
endo(x)
man(x)
lat(x)
```

Arguments

x	A "semPlotModel" object

Author(s)

Sacha Epskamp <mail@sachaepskamp.com>

See Also

semPlotModel

semPlotModel_S4-methods *S4 methods for semPlotModel*

Description

S4 generic used only for the lavaan-class class. See semPlotModel for more information and semPlotModel-class for the resulting object.

Methods

signature(object = "lavaan") A lavaan-class object.

`semSyntax` *Produce model syntax for various SEM software*

Description

This function produces a model object or model syntax for SEM software based on a `semPlotModel-class` object. If the input is not a `"semPlotModel"` object the `semPlotModel` function is run on the input. This allows to create model syntax for one program based on the output of another program.

Currently only the R packages 'lavaan' (Rosseel, 2012) and 'sem' (Fox, Nie & Byrnes, 2012) are supported.

Usage

```
semSyntax(object, syntax = "lavaan", allFixed = FALSE, file)
```

Arguments

`object`	A "semPlotModel" object or any of the input possibilities for `semPlotModel`.
`syntax`	A string indicating which syntax to be used for the output. Currently supported are `'lavaan'` and `'sem'`.
`allFixed`	Logical, should all parameters be fixed to their estimate. Useful for simulating data.
`file`	Path of a file the model should be written to.

Value

A string containing the `lavaan` model syntax or a `"semmod"` object for the `sem` package.

Author(s)

Sacha Epskamp <mail@sachaepskamp.com>

References

Yves Rosseel (2012). lavaan: An R Package for Structural Equation Modeling. Journal of Statistical Software, 48(2), 1-36. URL http://www.jstatsoft.org/v48/i02/.

John Fox, Zhenghua Nie and Jarrett Byrnes (2012). sem: Structural Equation Models. R package version 3.0-0. http://CRAN.R-project.org/package=sem

See Also

`semPlotModel` `semPlotModel-class` `semPaths`

Examples

```
# MIMIC model, example 5.8 from mplus user guide:
tryres <- try({
 Data <- read.table("http://www.statmodel.com/usersguide/chap5/ex5.8.dat")
})

if (!is(tryres,"try-error")){


names(Data) <- c(paste("y", 1:6, sep=""),
                 paste("x", 1:3, sep=""))

# Data <- Data[,c(7:9,1:6)]

# Model:
model.Lavaan <- 'f1 =~ y1 + y2 + y3
f2 =~ y4 + y5 + y6
f1 + f2 ~ x1 + x2 + x3 '

# Run Lavaan:
library("lavaan")
fit.Lavaan <- lavaan:::cfa(model.Lavaan, data=Data, std.lv=TRUE)

# Obtain Lavaan syntax:
model.Lavaan2 <- semSyntax(fit.Lavaan, "lavaan")

# Run Lavaan again:
fit.Lavaan2 <- lavaan:::lavaan(model.Lavaan2, data=Data)

# Compare models:
layout(t(1:2))
semPaths(fit.Lavaan,"std",title=FALSE)
title("Lavaan model 1",line=3)
semPaths(fit.Lavaan2, "std",title=FALSE)
title("Lavaan model 2",line=3)

# Convert to sem model:
model.sem <- semSyntax(fit.Lavaan, "sem")

# Run sem:
library("sem")
fit.sem <- sem:::sem(model.sem, data = Data)

# Compare models:
layout(t(1:2))
semPaths(fit.Lavaan,"std",title=FALSE)
title("Lavaan",line=3)
semPaths(fit.sem, "std",title=FALSE)
title("sem",line=3)
}
```

Index