Package 'semPLS'

February 20, 2015

Type Package

Title Structural Equation Modeling Using Partial Least Squares

Version 1.0-10

Date 2013-01-21

Author Armin Monecke <armin.monecke@stat.uni-muenchen.de>

Maintainer Armin Monecke <armin.monecke@stat.uni-muenchen.de>

Description Fits structural equation models using partial least squares (PLS). The PLS approach is referred to as 'soft-modeling' technique requiring no distributional assumptions on the observed data.

License GPL-2

Depends lattice (>= 0.20-5)

Suggests XML, boot, sem, matrixcalc

LazyLoad yes

Repository CRAN

Date/Publication 2013-01-21 15:09:52

NeedsCompilation no

R topics documented:

bootsempls *Bootstrap a PLS path model*

Description

Bootstraps a PLS path model in a `sempls` object (as returned by the `sempls` method).

Usage

```
bootsempls(object, nboot=200, start=c("ones", "old"), method=
            c("ConstructLevelChanges", "IndividualSignChanges",
            "Standard"), verbose=TRUE, strata, ...)

## S3 method for class 'bootsempls'
print(x, digits=3, ...)

## S3 method for class 'bootsempls'
summary(object, type=c("perc", "bca", "norm", "basic", "none"),
        level=0.95, ...)

## S3 method for class 'summary.bootsempls'
print(x, na.print, digits = 3, ...)

## S3 method for class 'bootsempls'
densityplot(x, data, pattern="beta", subset=NULL, ...)

## S3 method for class 'bootsempls'
parallelplot(x, data, pattern="beta", subset=NULL, reflinesAt,
         col=c("grey", "darkred", "darkred", "black"), lty=c("solid",
         "solid","dashed", "dotted"), ...)
```

Arguments

`object`	An object of class `sempls` as returned by the method `sempls`.
`nboot`	The number of bootstrap replications; the default is `200`.
`start`	A `character` value defining the initialisation of outer weights. If • `start="ones"`, then the outer weights for each block are initialised by ones. • `start="old"`, the final outer weights taken from the `sempls` object are used for initialisation.
`method`	A `character` value, which can take the values: • `"ConstructLevelChanges"` (default): The vector of loadings for each LV in each resample is compared to the corresponding vector of loadings in the original sample. The signs of the weights, and consequently the signs of the loadings, are reversed if the absolute value of the sum differences

between estimated loadings obtained from the original sample and the estimated loadings obtained from the resample is bigger than the absolute value of the sum sums of the latter, see Tenenhaus et al (2005).
- `"IndividualSignChanges"`: not implemented yet.
- `"Standard"`: No compensation for sign changes of resampled statistics.

`verbose`	A `logical` indicating, whether progress of bootstrap shall be displayed.
`x`	An object of class `bootsempls` and `summary.bootsempls` respectively.
`na.print`	A `character` substituting values not to be printed.
`digits`	Controls the number of digits to print.
`type`	Type of bootstrapped confidence intervals to compute; the default is `"perc"` (percentile); see `boot.ci` for details.
`strata`	An integer vector or factor specifying the strata for multi-sample problems. If the argument is not provided, all data is assumed to come from the same sample. For details, see `boot.ci`.
`level`	Level for confidence intervals; default is `0.95`.
`...`	Arguments to be passed down to other methods.
`data`	The `data` is not used because the `bootsempls` object already contains the required data.
`pattern`	A regular expression passed on to `grep`. It is helpful to easily identify a set of coefficients, e.g. `pattern="beta"` plots the path coefficients only.
`subset`	Index or character vector of coefficients to include. Note, that `subset` overrides `pattern`.
`reflinesAt`	A vector of values at which to plot reference lines into the parallel cooordinates.
`col`	Colors for bootstrap statistics, sample statistic, lower and upper bootstrap confidence levels and reference lines.
`lty`	Line type for bootstrap statistics, sample statistic, lower and upper bootstrap confidence levels and reference lines.

Details

`boot.sempls` implements the nonparametric bootstrap, assuming an independent random sample. Convergence failures in the bootstrap resamples are discarded (and a warning printed); 10 consecutive convergence failures result in an error. You can use the `boot` function in the boot package for more complex sampling schemes and additional options.

Value

`boot.sempls` returns an object of class `bootsempls`, which inherits from class `boot`, supported by the `boot` package. The returned object contains the following components:

`t0`	The estimated parameters in the model fit to the original data set.
`t`	a matrix containing the bootstrapped estimates, one bootstrap replication per row.
`data`	The data frame containing the data to which the model was fit.

seed	The value of .Random.seed when boot.sempls was called.
statistic	The function used to produce the bootstrap replications; this is always the local function refit from boot.sempls.
sim	Always set to "ordinary"; see the documentation for the boot function.
stype	Always set to "i"; see the documentation for the boot function.
call	The call of the boot.sem function.
tryErrorIndices	Contains the indices for each resample, which returned try-error.
clcIndices	When the method ConstructLevelChanges is used, it stores the indices of the blocks of MVs for which a sign change occured.
bootIndices	A matrix containing the indices of the converged bootstrap samples as rows.
outer_weights	A martrix containing, as rows, the outer weights for each bootsrap sample.
fitted_model	The fitted sempls model returned from sempls.
strata	The strata used. This is the vector passed to boot, if it was supplied or a vector of ones if there were no strata.

References

Tenenhaus, M., V. E. Vinzi, Y.-M. Chatelin, and C. Lauro (2005) PLS path modeling. Computational Statistics & Data Analysis 48, 159-205.

See Also

boot, boot.sem

Examples

```
## Not run:
data(ECSImobi)
ecsi <- sempls(model=ECSImobi, data=mobi)

### Bootstrapping
set.seed(123)
ecsiBoot <- bootsempls(ecsi, nboot=200, start="ones", verbose=TRUE)
summary(ecsiBoot, type="perc", level=0.95)



## inspectation of bootstrap samples
parallelplot(ecsiBoot, subset=1:ncol(ecsiBoot$t), reflinesAt=0)

# only inspecting the path coefficients
parallelplot(ecsiBoot, pattern="beta", reflinesAt=c(0,1))
densityplot(ecsiBoot, pattern="beta")

# only inspecting the outer loadings
parallelplot(ecsiBoot, pattern="lam")
```

```
# only inspecting the outer loadings for Loyalty
parallelplot(ecsiBoot, pattern="lam7", type="perc", level=0.90,
         main="Loyalty\n 200 bootstrapped outer loadings")

## End(Not run)
```

---

hanafi2007 *Hanafi (2007): Example Data*

Description

Example data used by Hanafi (2007) for the computation of latent variables with the estimation mode B.

Usage

```
data(hanafi2007)
```

Format

A data frame containing 10 observations of 11 variables.

References

Hanafi, M. (2007), PLS Path modelling: computation of latent variables with the estimation mode B, Computational Statistics 22, 275-292.

See Also

plsm, sempls

Examples

```
### get the Data from Hanafi's Example
data(hanafi2007)
hanafi2007

### 1st Example: the 11 MVs relate to formative 3 LVs
Ex1mfrom <- names(hanafi2007)
Ex1mto <- paste("z", c(rep(1:3, each=3), 3, 3), sep="")
Ex1mm <- cbind(source=Ex1mfrom, target=Ex1mto)

Ex1sfrom <- paste("z", 1:2, sep="")
Ex1sto <- paste("z", 2:3, sep="")
Ex1sm <- cbind(source=Ex1sfrom, target=Ex1sto)

library(semPLS)
EX1 <- plsm(data=hanafi2007, strucmod=Ex1sm, measuremod=Ex1mm)

ex1A <- sempls(model=EX1, data=hanafi2007, E="A", maxit=100, tol=1e-7)
```

```
# for the centroid scheme criterion f is used:
matplot(ex1A$Hanafi[,"iteration"], 2*ex1A$Hanafi[,"f"],
        type="b", lty=1, pch=3,
        xlim=c(0, 29), ylim=c(0, 3.5),
        main="Hanafi 2007: Example 1\ncentroid scheme",
        ylab="Criterion f", xlab="Iteration")

legend(x=17, y=0.5, legend="Lohm{\"o}ller's procedure", lty=1, pch=3)

# check the evolution of outer weights
plot(ex1A, xlim=c(0, 30))


# Now the same using factorial scheme:
ex1B <- sempls(model=EX1, data=hanafi2007, E="B", maxit=100, tol=1e-7)

# for the factorial scheme criterion g is used:
matplot(ex1B$Hanafi[,"iteration"], 2*ex1B$Hanafi[,"g"],
        type="b", lty=1, pch=3,
        xlim=c(0, 29), ylim=c(0, 3),
        main="Hanafi 2007: Example 1\nfactorial scheme",
        ylab="Criterion g", xlab="Iteration")

legend(x=17, y=0.5, legend="Lohm{\"o}ller's procedure", lty=1, pch=3)


### 2nd Example: the 11 MVs relate to 5 formative LVs
# renaming of the variables
names(hanafi2007) <- paste("x", rep(1:5, c(2, 2, 2, 2, 3)), c(rep(1:2, 5), 3), sep="")
Ex2mfrom <- names(hanafi2007)
Ex2mto <- paste("z", c(rep(1:5, each=2), 5), sep="")
Ex2mm <- cbind(source=Ex2mfrom, target=Ex2mto)

Ex2sfrom <- paste("z", rep(1:4, 4:1), sep="")
Ex2sto <- paste("z", c(2:5, 3:5, 4:5, 5), sep="")
Ex2sm <- cbind(source=Ex2sfrom, target=Ex2sto)

EX2 <- plsm(data=hanafi2007, strucmod=Ex2sm, measuremod=Ex2mm)

# this time only for the centroid scheme with criterion f:
ex2A <- sempls(model=EX2, data=hanafi2007, E="A", maxit=100, tol=1e-7)
matplot(ex2A$Hanafi[,"iteration"], 2*ex2A$Hanafi[,"f"],
        type="b", lty=1, pch=3,
        xlim=c(0, 10), ylim=c(0, 12),
        main="Hanafi 2007: Example 2\ncentroid scheme",
        ylab="Criterion f", xlab="Iteration")

legend(x=6, y=1.5, legend="Lohm{\"o}ller's procedure", lty=1, pch=3)

# check the evolution of outer weights
plot(ex2A, xlim=c(0, 10))
```

mobi *Measurement Instrument for the Mobile Phone Industry*

Description

The data set is used as measurement instrument for the european customer satisfaction index (ECSI) adapted to the mobile phone market, see Tenenhaus et al. (2005).

Usage

```
data(mobi)
data(ECSImobi)
data(ECSImm)
data(ECSIsm)
```

Format

The data frame mobi has 250 observations on the following 24 items. All the items are scaled from 1 to 10.

CUEX1 Expectations for the overall quality of "your mobile phone provider" at the moment you became customer of this provider

CUEX2 Expectations for "your mobile phone provider" to provide products and services to meet your personal need

CUEX3 How often did you expect that things could go wrong at "your mobile phone provider"

CUSA1 Overall satisfaction

CUSA2 Fulfillment of expectations

CUSA3 How well do you think "your mobile phone provider" compares with your ideal mobile phone provider?

CUSCO You complained about "your mobile phone provider" last year. How well, or poorly, was your most recent complaint handled or You did not complain about "your mobile phone provider" last year. Imagine you have to complain to "your mobile phone rovider" because of a bad quality of service or product. To what extent do you think that "your mobile phone provider" will care about your complaint?

CUSL1 If you would need to choose a new mobile phone provider how likely is it that you would choose "your provider" again?

CUSL2 Let us now suppose that other mobile phone providers decide to lower their fees and prices, but "your mobile phone provider" stays at the same level as today. At which level of difference (in %) would you choose another mobile phone provider?

CUSL3 If a friend or colleague asks you for advice, how likely is it that you would recommend "your mobile phone provider"?

IMAG1 It can be trusted what it says and does

IMAG2 It is stable and firmly established

IMAG3 It has a social contribution to society

IMAG4 It is concerned with customers

IMAG5 It is innovative and forward looking

PERQ1 Overall perceived quality

PERQ2 Technical quality of the network

PERQ3 Customer service and personal advice offered

PERQ4 Quality of the services you use

PERQ5 Range of services and products offered

PERQ6 Reliability and accuracy of the products and services provided

PERQ7 Clarity and transparency of information provided

PERV1 Given the quality of the products and services offered by "your mobile phone provider" how would you rate the fees and prices that you pay for them?

PERV2 Given the fees and prices that you pay for "your mobile phone provider" how would you rate the quality of the products and services offered by "your mobile phone provider"?

Details

The data frame mobi contains the observed data for the model specified by ECSImobi. The from-to-matrizes ECSImm and codeECSIsm represent directed edges connecting the variables in outer/measurement and inner/structural model. They are needed for model specification by mean of the plsm method.

References

Tenenhaus, M., V. E. Vinzi, Y.-M. Chatelin, and C. Lauro (2005) PLS path modeling. Computational Statistics & Data Analysis 48, 159-205.

See Also

ECSImobi, ECSIsm, ECSImm

Examples

```
data(mobi)
data(ECSImobi)
ecsi <- sempls(model=ECSImobi, data=mobi, E="C")
ecsi
```

---

pathDiagram *Draw Path Diagram*

Description

pathDiagram creates a description of the path diagram for a PLS path model object of class sempls to be processed by the graph-drawing program *dot*; see Gansner, Koutsofios, and North (2006) and http://www.graphviz.org/.

Usage

```
pathDiagram(object, ...)


## S3 method for class 'sempls'
pathDiagram(object, file, min.rank=NULL, max.rank=NULL, same.rank=NULL,
            edge.labels=c("names", "values", "both"), size=c(8,8),
            node.font=c("Helvetica", 14), edge.font=c("Helvetica", 10),
            rank.direction=c("LR", "TB"), digits=2, output.type =
            c("graphics", "dot"), graphics.fmt = "pdf", dot.options=NULL,
            rSquared=NULL, full=TRUE, ...)
```

Arguments

`object`	A `sempls` object
`...`	Arguments to be passed down to `pathDiagram.sempls`.
`file`	A file in which to write the *dot* description of the path diagram; if not specified, the description is written to standard output (normally the R console).
`min.rank`	A `character` string listing names of variables to be assigned minimum rank (order) in the graph; the names should be separated by commas.
`max.rank`	A `character` string listing names of variables to be assigned maximum rank in the graph; the names should be separated by commas.
`same.rank`	A `character` string or vector of `character` strings of variables to be assigned equivalent rank in the graph; names in each string should be separated by commas.
`edge.labels`	`"names"` to label arrows with parameter names; `"values"` to label arrows with parameter estimates, or `both`
`size`	The size of the graph, in inches.
`node.font`	Font name and point-size for printing variable names.
`edge.font`	Font name and point-size for printing arrow names or values.
`rank.direction`	Draw graph left-to-right, `"LR"`, the default, or top-to-bottom, `"TB"`.
`digits`	Number of digits after the decimal point (default, 2) to which to round parameter estimates.
`output.type`	If `"graphics"`, the default, both a ".dot" file and a graphics file will be created.
`graphics.fmt`	A graphics format recognized by the dot program; the default is "pdf"; `graphics.fmt` is also used for the extension of the graphics file that is created.
`dot.options`	Options to be passed to the *dot* program, given as a `character` string.
`rSquared`	A matrix returned from method `rSquared`. If supplied, the R-squared values are printed in the diagram.
`full`	If `FALSE` the .dot file is created only for the structural model.

Value

`pathDiagram` is used for its side-effect, producing a graph description of the model.

References

Gansner, E., E. Koutsofios, and S. North (2006) Drawing graphs with *dot*. http://www.graphviz.org/Documentation/dotguide.pdf.

See Also

sempls, rSquared

Examples

```
### create .pdf file for the path diagram
### Note that graphviz (www.graphviz.org) must be available.
## Not run:
pathDiagram(ecsi, file="ecsiPLS1", edge.labels="both",
            output.type="graphics", digits=3, graphics.fmt = "pdf")

# include R-squared values
rSquared <- rSquared(ecsi)
pathDiagram(ecsi, file="ecsiPLS2", edge.labels="both",
            output.type="graphics", digits=3, graphics.fmt = "pdf",
            rSquared=Rsquared)

# only the structural model
pathDiagram(ecsi, file="ecsiPLS3", edge.labels="both",
            output.type="graphics", digits=3, graphics.fmt = "pdf",
            rSquared=Rsquared, full=FALSE)

## End(Not run)
```

---

plsm *Specification of Path Model*

Description

Creates the specification of a path model used by sempls. The structural and the measurement must be specified in .csv files. The paths must be entered in the form of a from-to matrix. The variable in the first column represents the source and the second column represents the target of each path.

Usage

```
plsm(data, strucmod, measuremod, order=c("generic",
     "alphabetical"), interactive=FALSE)

mvplot(model, ...)
## S3 method for class 'plsm'
mvplot(model, data, LVs, ask=TRUE, ...)

mvpairs(model, ...)
## S3 method for class 'plsm'
mvpairs(model, data, LVs, ask=TRUE, ...)
```

Arguments

data	A data.frame intendend to use for the fitting method, sempls.
strucmod	Either a from-to-matrix representing the inner/structural model or the path to an .csv file representing the inner/structural model.
measuremod	Either from-to-matrix representing the outer/measurement models or the path to an .csv file representing the outer/measurement models.
order	A character describing how to order the latent variables (LVs). If • "generic" the LVs will be ordered according to their appearance in the causal chain of the structural model (default). • "alphabetical" the LVs will be ordered alphabetically.
interactive	Logical indicating whether to specify the model interactively using edit. The default is FALSE
model	An object of class plsm.
LVs	A character vector naming the blocks of LVs for which to create the plots.
ask	See ?par
...	Arguments to pass down to other methods, e.g., par.

Value

The object returned is of class plsm with the elements:

latent	A character vector naming the latent variables.
manifest	A character vector naming the manifest variables.
strucmod	Contains only the subset of path representing the structural model.
measuremod	Contains only the subset of path representing the measurement model.
D	The adjacency matrix $\boldsymbol{D}$ for the structural model.
M	The adjacency matrix $\boldsymbol{M}$ for the measurement model.
blocks	A list naming the MVs belonging to each LV's block and telling their measurement mode.
order	See armguments section.

See Also

sempls, read.splsm

Examples

```
# getting the path to the .csv file representing the inner Model
ptf_Struc <- system.file("ECSIstrucmod.csv", package="semPLS")

# getting the path to the .csv file representing the outer Models
ptf_Meas <- system.file("ECSImeasuremod.csv", package="semPLS")
```

```
sm <- as.matrix(read.csv(ptf_Struc))
mm <- as.matrix(read.csv(ptf_Meas))
data(mobi)

ECSI <- plsm(data=mobi, strucmod=sm, measuremod=mm)

# Adjacency matrix of the structural model
ECSI$D

# Adjacency matrix of the measurement model
ECSI$M

# return all elements
ECSI

### Interactive mode
## Not run:
# specify model in a spreadsheets
ECSI <- plsm(data=mobi, interactive=TRUE)
ECSI

## End(Not run)

### explore blocks of MVs
mvplot(model=ECSI, data=mobi, LVs="Expectation")
mvpairs(model=ECSI, data=mobi, LVs="Expectation")
```

---

plsm2sem *Convert Model for Use in* **sem**

Description

Converts a plsm object to an object of class mod for usage of sem method within **sem** package.

Usage

```
plsm2sem(model, ...)

## S3 method for class 'plsm'
plsm2sem(model, file=stdout(), fixedVarMV=TRUE, fixedVarLV=TRUE,
          fixedLoad=character(), ...)
```

Arguments

model	An object of class splsm as returned by the method read.splsm.
...	Arguments to pass down.
file	A character naming the file to write to. If no file argument is specified, splsm2sem writes to stdout.

fixedVarMV	A logical indicating whether the variances of the MVs should be fixed to one. The default is TRUE.
fixedVarLV	A logical indicating whether the variances of the LVs should be fixed to one. The default is TRUE.
fixedLoad	A character vector naming the MVs, for which the outer loading should be fixed to one.

Value

If the sem package is available, an object of class mod (see specifyModel) is returned, else a text representation of recticular action model (RAM) is written. Note, specifyModel was specify.model in **sem (< 2.0.0)**.

See Also

specifyModel

Examples

```
data(ECSImobi)
if(require(sem)){
   ECSIsem <- plsm2sem(ECSImobi)
   detach(package:sem)
   ECSIsem
}
```

---

plsmUtils *Utility methods for* plsm *objects.*

Description

By means of the utility methods pls model descrpitions inheriting from class plsm can easily be altered or investigated.

Usage

```
plsmEdit(model, ...)
## S3 method for class 'plsm'
plsmEdit(model, data, ...)

addLV(model, ...)
## S3 method for class 'plsm'
addLV(model, data, LV, mode, MVs, pred, succ, ...)

invertLVs(model, ...)
## S3 method for class 'plsm'
invertLVs(model, LVs, ...)
```

```
removeLVs(model, ...)
## S3 method for class 'plsm'
removeLVs(model, which, ...)

addMVs(model, ...)
## S3 method for class 'plsm'
addMVs(model, data, LV, MVs, ...)

removeMVs(model, ...)
## S3 method for class 'plsm'
removeMVs(model, MVs, ...)

addPath(model, ...)
## S3 method for class 'plsm'
addPath(model, from, to, ...)

removePath(model, ...)
## S3 method for class 'plsm'
removePath(model, from, to, ...)

exogenous(model)
endogenous(model)
reflective(model)
formative(model)
indicators(model, LV)
predecessors(model)
successors(model)
connected(model)
acyclic(model)
```

Arguments

model	An object inheriting from class plsm as returned from plsm or read.splsm.
data	A data.frame containing the observed variables (MVs). The storage mode for all the MVs included in the model must be numeric.
LV	A character value naming an LV to add or the LV to associate with newly created MVs.
mode	A character value, specifying the measurement mode of the LV to add: • "A" for a reflective LV, • "B" for a formative LV.
MVs	A character vector naming MVs, which are either to be added, deleted or associated with a newly created LV. Note: the names of the MVs must exist as numeric variables in data.
pred	A character vector naming the predecessors of an LV. If pred is NULL the LV is exogenous.
succ	A character vector naming the successors of an LV.

LVs	A character vector naming LVs to invert. Invert means to invert the direction of pathes of associated measurement models. This means changing reflective measurement to formative and vice versa.
which	A character vector naming LVs to delete.
from	A character vector naming predecessors of LVs specified in to.
to	A character vector naming successors of LVs specified in from.
...	Currently unused.

Details

exogenous returns the exogenous latent variables contained in a path model.

endogenous returns the endogenous latent variables contained in a path model.

reflective returns the reflectively measured latent variables contained in a path model.

formative returns the formatively measured latent variables contained in a path model.

indicators returns the index or observed variables related to the given latent variable.

predecessors returns a list with the names of predecessors for each latent variable.

successors returns a list with the names of successors for each latent variable.

connected returns TRUE if the inner model is a connected graph.

acyclic returns TRUE if the inner model is recursive, thus represented by an acyclic graph.

Value

All utility methods return an object of class plsm.

See Also

plsm, read.splsm

Examples

```
data(mobi)
data(ECSIsm)
data(ECSImm)

ECSI <- plsm(data=mobi, strucmod=ECSIsm, measuremod=ECSImm)


### Some sense free examples

# Print the block of MVs associated with "Expectation".
ECSI[["blocks"]]["Expectation"]

# Change measurement model for "Expectation" from reflective to
# formative and print its block MVs
invertLVs(model=ECSI, LVs=c("Expectation"))[["blocks"]]["Expectation"]

# Print the adjacency matrix for the inner model.
```

```
ECSI[["D"]]

# Add an additional path from "Quality" to "Loyalty" and
# print the resulting adjacency matrix.
addPath(model=ECSI, from="Quality", to="Loyalty")[["D"]]

# Remove the path previously added.
removePath(model=ECSI, from="Image", to=c("Satisfaction", "Loyalty"))[["D"]]

# Print all MVs used in the model
ECSI$manifest

# Remove some MVs and print the MVs used in the model
removeMVs(model=ECSI, MVs=c("IMAG3", "CUEX1", "PERQ7"))$manifest


### some handy functions

exogenous(ECSI)
endogenous(ECSI)
reflective(ECSI)
formative(ECSI)
indicators(ECSI, "Image")
predecessors(ECSI)
```

`read.splsm` *Import of XML Model Description Specified in SmartPLS*

Description

Imports a .splsm file, an XML model description, specified in SmartPLS (see references). Note, that moderator effects specified within SmartPLS are not yet supported. Nevertheless moderating effects can be specified manually.

Usage

```
read.splsm(file=character(), order=c("generic", "alphabetical"))
```

Arguments

`file`	A `character` naming the path to the `.splsm` file produced by SmartPLS.
`order`	A `character` describing how to order the latent variables (LVs). If • `"generic"` the LVs will be ordered according to their appearance in the causal chain of the structural model (default). • `"alphabetical"` the LVs will be ordered alphabetically.

Value

The object returned is of class `splsm` with the elements:

`connectionIDs`	A `data.frame` containing a source id and target id for all connections of the path model, as specified in the `.splsm` file produced by SmartPLS. The ids refer to the nodes of latent and manifest variable.
`variables`	A `data.frame` containing the names of all the nodes together with their id and x -, y-position on the SmartPLS GUI.
`latent`	A character vector naming the latent variables.
`manifest`	A `character` vector naming the manifest variables.
`path`	Is somehow identical with `connectionIDs`, only that the node ids are replaced by their names.
`strucmod`	Contains only the subset of `path` representing the structural model.
`measuremod`	Contains only the subset of `path` representing the measurement model.
`D`	The adjacency matrix $\boldsymbol{D}$ for the structural model.
`M`	The adjacency matrix $\boldsymbol{M}$ for the measurement model.
`blocks`	A `list` naming the MVs belonging to each LV's block and telling their measurement mode.
`order`	See armguments section.

References

Ringle, C.M./Wende, S./Will, S.: SmartPLS 2.0 (M3) Beta, Hamburg 2005, http://www.smartpls.de.

See Also

`sempls`, `plsm`

Examples

```
# getting the path to file: 'ECSI_Tenenhaus.splsm' (generated by SmartPLS)
ptf <- system.file("SmartPLS", "workspace", "ecsi", "ECSI_Tenenhaus.splsm",
                   package="semPLS")

# creating the model specification to use with 'sempls()'.
ECSI <- read.splsm(ptf)
ECSI
```

sempls *Structural Equation Modeling by Partial Least Squares*

Description

sempls fits structural equation models by the patial least squares (PLS) method. The estimation is based on the raw data and requires no distributional assumptions.

Usage

```
sempls(model, ...)

## S3 method for class 'plsm'
sempls(model, data, maxit=20, tol=1e-7,
         scaled=TRUE, sum1=FALSE, wscheme="centroid", pairwise=FALSE,
         method=c("pearson", "kendall", "spearman"),
         convCrit=c("relative", "square"),
         verbose=TRUE, ...)

## S3 method for class 'sempls'
print(x, digits=2, ...)
## S3 method for class 'sempls'
plot(x, ...)
## S3 method for class 'sempls'
densityplot(x, data, use=c("fscores", "prediction",
            "residuals"), ...)

pathCoeff(object, ...)
## S3 method for class 'sempls'
pathCoeff(object, ...)
## S3 method for class 'pathCoeff'
print(x, na.print=".", digits=2, abbreviate=FALSE, ...)

totalEffects(object)
## S3 method for class 'sempls'
totalEffects(object)
## S3 method for class 'totalEffects'
print(x, na.print=".", digits=2, abbreviate=FALSE,...)

plsWeights(object)
## S3 method for class 'sempls'
plsWeights(object)
## S3 method for class 'plsWeights'
print(x, na.print=".", digits=2, abbreviate=FALSE, ...)

plsLoadings(object)
## S3 method for class 'sempls'
```

```
plsLoadings(object)
## S3 method for class 'plsLoadings'
print(x, type=c("discriminant", "outer", "cross"),
     cutoff=NULL, reldiff=0.2, na.print=".", digits=2, abbreviate=FALSE, ...)
```

Arguments

model	An object inheriting from class plsm as returned from plsm or read.splsm.
...	Arguments to be passed down.
data	A data.frame containing the observed variables (MVs). The storage mode for all the MVs included in the model must be numeric.
maxit	A numeric value, which determines the maximum number of iterations performed by the PLS algorithm. The default is 20 iterations.
tol	A numeric value, specifying the tolerance for the maximum relative differences in the outer weights. The default value is 10^{-7}.
scaled	A logical value indicating, whether the observed data shall be scaled to zero mean and unit variance. The default is TRUE.
sum1	A logical value indicating, whether the outer weights foreach latent variable (LV) shall be standardized to sum up to one. The default is FALSE. Since the factor scores are scaled in each step of the PLS algorithm, changing this value to TRUE does not affect the results.
wscheme	A character naming the weighting scheme to use. Possible values are: • "A" or "centroid" for the centroid scheme, the default, • "B" or "factorial"for the factorial scheme and • "C", "pw" or "pathWeighting" for the path weighting scheme.
pairwise	A logical value indicating, whether correlations shall be calculated pairwise. If the observed data does not contain missing values, the results are not affected. The default is FALSE. For more details the R help, ?cor, can be consulted.
method	A character naming the method to calculate the correlations. Possible values are: • "pearson" , the default, • "kendall", • "spearman". For more details on the method, the R help, ?cor, can be consulted. Note, that despite of the method argument, pearson correlations are always used for the inner approximation (step 2).
convCrit	The convergence criteria to use: • "relative", the default, • "square".
verbose	Logical: If FALSE no status messages are printed.
object	An object of class sempls.
x	An object of the according class.

type	If the argument what="loadings", type describes the loadings to be extracted – those are: • "discriminant", the default, contrasts outer against cross loadings to check for discrimant validity of the measurement model, • "outer" for the outer loadings and • "cross" for the cross loadings.
cutoff	A numerical value at which to cutoff the loadings – this means loadings smaller than the cutoff value will not be printed.
reldiff	The argument is only effectiv when type="discriminant". It is a numeric value, specifying the relative difference between outer and cross loadings at which cross loadings will still be printed.
na.print	A character substituting values not to be printed.
digits	minimal number of _significant_ digits, see print.default.
use	The values for which the density plots are created. If • "fscores": the factor scores are used, • "prediction": the estimated factor scores are used, • "residuals": the residuals are used.
abbreviate	A logical indicating whether dimnames should be abbreviated. For Details see abbreviate. The default is FALSE.

Value

sempls returns an object of class sempls, with the following elements:

coefficients	A data.frame containing the estimates for all the arcs in the path model, those are the outer loadings for mode 'A' type LVs and outer weights for mode 'B' type LVs and path coefficients for those belonging to the structural model.
path_coefficient	The matrix of path coefficients.
outer_loadings	The matrix of outer loadings.
cross_loadings	The matrix of cross loadings.
total_effects	The matrix of total effects.
inner_weights	The matrix of inner weights.
outer_weights	The matrix of outer weights.
factor_scores	A data.frame containing the estimated factor scores for the LVs.
data	A data.frame containing the preprocessed obseravtions of the MVs.
incomplete	The index of the incomplete observations.
...	All the other values are just storing information used in the call.

See Also

plsm, read.splsm, rSquared, pathDiagram, bootsempls, plsm2sem, sem

Examples

```
data(ECSImobi)
ecsi <- sempls(model=ECSImobi, data=mobi, wscheme="pathWeighting")
ecsi

## create plots
densityplot(ecsi)
densityplot(ecsi, use="prediction")
densityplot(ecsi, use="residuals")

## Values of 'sempls' objects
names(ecsi)
ecsi$outer_weights
ecsi$outer_loadings
ecsi$path_coefficients
ecsi$total_effects


### using convenience methods to sempls results
## path coefficients
pathCoeff(ecsi)

## total effects
totalEffects(ecsi)

## get loadings and check for discriminant validity
(l <- plsLoadings(ecsi))
# outer loadings
print(l, type="outer", digits=2)
# outer loadings greater than 0.5
print(l,type="outer", cutoff=0.5, digits=2)
# cross loadings greater than 0.5
print(l, type="cross", cutoff=0.5, digits=2)


### R-squared
rSquared(ecsi)


### Create .dot representation of the path diagram and
### create .pdf file if graphviz is available.
## Not run:
pathDiagram(ecsi, file="ecsiPLS1", edge.labels="both",
            output.type="graphics", digits=3, graphics.fmt = "pdf")

# include R-squared values
pathDiagram(ecsi, file="ecsiPLS2", edge.labels="both",
            output.type="graphics", digits=3, graphics.fmt = "pdf",
            rSquared=rSquared(ecsi))

# only the structural model
pathDiagram(ecsi, file="ecsiPLS3", edge.labels="both",
```

```
            output.type="graphics", digits=3, graphics.fmt = "pdf",
            rSquared=rSquared(ecsi), full=FALSE)

## End(Not run)
```

---

semplsGOF *Quality Indices and Goodness of fit measures for pls path models*

Description

A collection of method to validate the goodness of the model. Since there is no well identified global optimization criterion each part of the model needs to be validated.

Usage

```
rSquared(object, ...)
## S3 method for class 'sempls'
rSquared(object, na.rm=FALSE, ...)
## S3 method for class 'rSquared'
print(x, na.print=".", digits=2, ...)

qSquared(object, ...)
## S3 method for class 'sempls'
qSquared(object, d=NULL, impfun, dlines=TRUE,
         total=FALSE, ...)
## S3 method for class 'qSquared'
print(x, na.print=".", digits=2, ...)

dgrho(object, ...)
## S3 method for class 'sempls'
dgrho(object, ...)
## S3 method for class 'dgrho'
print(x, na.print=".", digits=2, ...)

communality(object, ...)
## S3 method for class 'sempls'
communality(object, ...)
## S3 method for class 'communality'
print(x, na.print=".", digits=2, ...)

redundancy(object, ...)
## S3 method for class 'sempls'
redundancy(object, ...)
## S3 method for class 'redundancy'
print(x, na.print=".", digits=2, ...)

rSquared2(object, ...)
```

```
## S3 method for class 'sempls'
rSquared2(object, na.rm=FALSE, ...)
## S3 method for class 'rSquared2'
print(x, na.print=".", digits=2, ...)

gof(object, ...)
## S3 method for class 'sempls'
gof(object, ...)
## S3 method for class 'gof'
print(x, na.print=".", digits=2, ...)
```

Arguments

object	An object of class sempls.
d	A numeric value for the omission distance. Thus choosing $d = N$, where N is the number of complete observations, is leaving one out cross validation. This is done when d takes its default value NULL.
impfun	An user specified function to impute missing values.
dlines	If TRUE the same observations are deleted for a whole block of MVs, else each dth , counting from top left to bottom right, observation is deleted.
total	If total=TRUE total effects are used instead of path coefficients to calculate the predictions.
na.rm	If na.rm=TRUE observation with missing values are discarded before analysis.
x	An object of the according class.
na.print	A character substituting values not to be printed.
digits	minimal number of _significant_ digits, see print.default.
...	Arguments to be passed down.

Value

Most GOF methods return a column vector with the names of the variables as rows and the respective measure as column.

References

Esposito Vinzi V., Trinchera L., Amato S. (2010). PLS Path Modeling: From Foundations to Recent Developments and Open Issues for Model Assessment and Improvement. In Esposito Vinzi V., Chin W.W., Henseler J., Wang H.F. (eds.), Handbook of Partial Least Squares: Concepts, Methods and Applications in Marketing and Related Fields, chapter 2. Springer-Verlag Berlin Heidelberg.

See Also

sempls, plsLoadings

Examples

```
data(ECSImobi)
ecsi <- sempls(model=ECSImobi, data=mobi, E="C")

### R-squared
rSquared(ecsi)

### Q-squared with omission distance d=4
qSquared(ecsi, d=4)

### Dillon-Goldstein's rho (aka composite reliability)
dgrho(ecsi)

### Communalities
communality(ecsi)

### Redundancy
redundancy(ecsi)

### R-squared (normal + corrected)
rSquared2(ecsi)

### Goodness of fit
gof(ecsi)

### check for discriminant validity using loadings
l <-plsLoadings(ecsi)
print(l, type="discriminant", cutoff=0.5, reldiff=0.2)
```

Index