# Package 'plspm'

July 23, 2025

**Type** Package

**Title** Partial Least Squares Path Modeling (PLS-PM)

**Version** 0.5.1

**Date** 2024-01-23

**Description** Partial Least Squares Path Modeling (PLS-PM), Tenenhaus, Esposito Vinzi, Chatelin, Lauro (2005) <doi:10.1016/j.csda.2004.03.005>, analysis for both metric and non-metric data, as well as REBUS analysis, Esposito Vinzi, Trinchera, Squillacciotti, and Tenenhaus (2008) <doi:10.1002/asmb.728>.

**URL** `https://github.com/gastonstat/plspm`

**BugReports** `https://github.com/gastonstat/plspm/issues`

**Depends** R (>= 3.0.1)

**Imports** tester, turner, diagram, shape, amap, methods

**Suggests** FactoMineR, ggplot2, reshape, testthat, knitr

**VignetteBuilder** knitr

**License** GPL-3

**LazyLoad** yes

**Collate** 'plspm.R' 'auxiliar.R' 'check_arguments.R' 'check_specifications.R' 'get_alpha.R' 'get_ave.R' 'get_boots.R' 'get_dummies.R' 'get_effects.R' 'get_generals.R' 'get_gof.R' 'get_inner_summary.R' 'get_manifests.R' 'get_metric.R' 'get_nom_scale.R' 'get_num_scale.R' 'get_ord_scale.R' 'get_path_scheme.R' 'get_paths.R' 'get_plsr1.R' 'get_PLSRdoubleQ.R' 'get_rank.R' 'get_rho.R' 'get_scores.R' 'get_treated_data.R' 'get_unidim.R' 'get_weights.R' 'get_weights_nonmetric.R' 'innerplot.R' 'outerplot.R' 'plot.plspm.R' 'rescale.R' 'summary_plspm.R' 'test_manifest_scaling.R' 'test_null_weights.R' 'unidimensionality.R' 'test_factors.R' 'russett-data.R' 'plspm.fit.R' 'plspm.groups.R' 'test_dataset.R' 'get_GQI.R' 'get_locals_test.R' 'get_scaled_data.R' 'it.reb.R' 'local.models.R' 'print.rebus.R' 'rebus.pls.R' 'rebus.test.R' 'res.clus.R' 'get_PLSR.R' 'get_PLSR_NA.R' 'quantiplot.R' 'plspm-package.R'

**NeedsCompilation** no

**Author** Frederic Bertrand [cre] (ORCID:
<https://orcid.org/0000-0002-0837-8281>),
Gaston Sanchez [aut],
Laura Trinchera [aut],
Giorgio Russolillo [aut]

**Maintainer** Frederic Bertrand <frederic.bertrand@utt.fr>

**Encoding** UTF-8

**Repository** CRAN

**Date/Publication** 2024-01-23 19:20:02 UTC

# Contents

`alpha` *Cronbach's alpha*

### Description

Cronbach's alpha of a single block of variables

### Usage

```
alpha(X)
```

### Arguments

| | |
|---|---|
| `X` | matrix representing one block of manifest variables |

### Value

Cronbach's alpha

### Author(s)

Gaston Sanchez

### See Also

`rho`, `unidim`

### Examples

```
## Not run:
 # load dataset satisfaction
 data(satisfaction)

 # block Image (first 5 columns of satisfaction)
 Image = satisfaction[,1:5]

 # compute Cronbach's alpha for Image block
 alpha(Image)

## End(Not run)
```

`arizona` *Arizona vegetation dataset*

## Description

This dataset gives the measurements of 16 vegetation communitites in the Santa Catalina Mountains, Arizona. The measurements were taken along different elevations from fir forest at high elevations, through pine forest, woodlands, and desert grassland.

## Format

A data frame with 16 observations and 8 variables. The variables refer to three latent concepts: 1) `ENV`=environment, 2) `SOIL`=soil, and 3) `DIV`=diversity.

| *Num* | *Variable* | *Description* | *Concept* |
|---|---|---|---|
| 1 | env.elev | Elevation (m) | environment |
| 2 | env.incli | Terrain inclination (degrees) | environment |
| 3 | soil.ph | Acidity and base saturation | soil |
| 4 | soil.orgmat | Organic matter content (perc) | soil |
| 5 | soil.nitro | Nitrogen content (perc) | soil |
| 6 | div.trees | Number of species of trees | diversity |
| 7 | div.shrubs | Numer of species of shrubs | diversity |
| 8 | div.herbs | Number of species of herbs | diversity |

The complete name of the rows are: 1) Abies lasiocarpa, 2) Abies concolor, 3) Pseudotsuga menziesii-Abies Concolor, 4) Pseudotsuga menziesii, 5) Pinus ponderosa-Pinus strobiformis, 6) Pinus ponderosa, 7) Pinus ponderosa-Quercus, 8) Pinus chihuahuana, 9) Pygmy conifer-oak scrub, 10) Open oak woodland, 11) Bouteloua curtipendula, 12) Spinose-suffrutescent, 13) Cercidium microphyllum, 14) Larrea divaricata, 15) Cercocarpus breviflorus, 16) Populus tremuloides.

## Source

Mixed data from Whittaker *et al* (1968), and Whittaker and Niering (1975). See **References** below.

## References

Whittaker, R. H., Buol, S. W., Niering, W. A., and Havens, Y. H. (1968) A Soil and Vegetation Pattern in the Santa Catalina Mountains, Arizona. *Soil Science*, **105**, pp. 440-450.

Whittaker, R. H., and Niering, W. A. (1975) Vegetation of the Santa Catalina Mountains, Arizona. V. Biomass, Production, and Diversity Along the Elevation Gradient. *Ecology*, **56**, pp. 771-790.

## Examples

```
data(arizona)
arizona
```

cereals *Cereals datset*

## Description

Data with several variables of different brands of cereal

## Usage

```
data(cereals)
```

## Format

A data frame with 77 observations on the following 15 variables.

- `mfr` Manufacturer of cereal
- `type` type: cold or hot
- `calories` calories per serving
- `protein` grams of protein
- `fat` grams of fat
- `sodium` milligrams of sodium
- `fiber` grams of dietary fiber
- `carbo` grams of complex carbohydrates
- `sugars` grams of sugars
- `potass` milligrams of potassium
- `vitamins` vitamins and minerals - 0, 25, or 100, indicating the typical percentage of FDA recommended
- `shelf` display shelf (1, 2, or 3, counting from the floor)
- `weight` weight in ounces of one serving
- `cups` number of cups in one serving
- `rating` a rating of the cereals

## Source

https://dasl.datadescription.com/datafile/cereals/

## Examples

```
# load data
data(cereals)

# take a peek
head(cereals)
```

college *College datasets*

## Description

Dataset with different scores (high school, undergrad basic, undergrad intermediate, and GPA) of graduated college student in life sciences majors

## Usage

```
data(college)
```

## Format

A data frame with 352 students on the following 13 variables. The variables may be used to construct four suggested latent concepts: 1) `HighSchool`=High School related scores, 2) `Basic`=scores of basic courses, 3) `InterCourse`=Scores of intermediate courses, 4) `GPA`=Final GPA (Graduate Point Average)

| *Num* | *Variable* | *Description* | *Concept* |
|---|---|---|---|
| 1 | `HS_GPA` | High School GPA | `HighSchool` |
| 2 | `SAT_Verbal` | Verbal SAT score | `HighSchool` |
| 3 | `SAT_Math` | Math SAT score | `HighSchool` |
| 4 | `Biology1` | Introductory Biology | `BasicCourses` |
| 5 | `Chemistry1` | Introductoy Chemistry | `BasicCourses` |
| 6 | `Math1` | Calculus 1 | `BasicCourses` |
| 7 | `Physics1` | Introductory Physics | `BasicCourses` |
| 8 | `Biology2` | Intermediate Biology | `InterCourses` |
| 9 | `Chemistry2` | Intermediate Chemistry | `InterCourses` |
| 10 | `Math2` | Calculus 2 | `InterCourses` |
| 11 | `Physics2` | Intermediate Physics | `InterCourses` |
| 12 | `FinalGPA` | Graduation GPA | `FinalGPA` |
| 13 | `Gender` | Gender | `none` |

## Examples

```
# load data
data(college)

# take a peek
head(college)
```

futbol — *Futbol dataset from Spain-England-Italy*

## Description

This data set contains the results of the teams in the Spanish, English, and Italian football leagues 2009-2010 season.

## Usage

```
data(futbol)
```

## Format

A data frame with 60 observations on the following 12 variables. The variables may be used to construct three latent concepts: 1) `ATTACK`=Attack, 2) `DEFENSE`=Defense, 3) `SUCCESS`=Success.

| *Num* | *Variable* | *Description* | *Concept* |
|---|---|---|---|
| 1 | GSH: Goals Scored at Home | total number of goals scored at home | ATTACK |
| 2 | GSA: Goals Scored Away | total number of goals scored away | ATTACK |
| 3 | SSH: Success to Score at Home | percentage of matches with scores goals at home | ATTACK |
| 4 | SSA: Success to Score Away | percentage of matches with scores goals away | ATTACK |
| 5 | NGCH: Goals Conceded at Home | total number (negative) of goals conceded at home | DEFENSE |
| 6 | NGCA: Goals Conceded Away | total number (negative) of goals conceded away | DEFENSE |
| 7 | CSH: Clean Sheets at Home | percentage of matches with no conceded goals at home | DEFENSE |
| 8 | CSA: Clean Sheets Away | percentage of matches with no conceded goals away | DEFENSE |
| 9 | WMH: Won Matches at Home | total number of matches won at home | SUCCESS |
| 10 | WMA: Won Matches Away | total number of matches won away | SUCCESS |
| 11 | Country: Leangue Country | country of the team's league | none |
| 12 | Rank: Rank Position | final ranking position within its league | none |

## Source

League Day.

## Examples

```
data(futbol)
futbol
```

innerplot *Plot inner model*

## Description

Plot the inner (structural) model for objects of class "plspm", as well as path matrices

## Usage

```
innerplot(x, colpos = "#6890c4BB", colneg = "#f9675dBB",
  box.prop = 0.55, box.size = 0.08, box.cex = 1,
  box.col = "gray95", lcol = "gray95", box.lwd = 2,
  txt.col = "gray50", shadow.size = 0, curve = 0,
  lwd = 3, arr.pos = 0.5, arr.width = 0.2, arr.lwd = 3,
  cex.txt = 0.9, show.values = FALSE, ...)
```

## Arguments

| | |
|---|---|
| x | Either a matrix defining an inner model or an object of class "plspm". |
| colpos | Color of arrows for positive path coefficients. |
| colneg | Color of arrows for negative path coefficients. |
| box.prop | Length/width ratio of ellipses. |
| box.size | Size of ellipses. |
| box.cex | Relative size of text in ellipses. |
| box.col | fill color of ellipses, |
| lcol | border color of ellipses. |
| box.lwd | line width of the box. |
| txt.col | color of text in ellipses. |
| shadow.size | Relative size of shadow of label box. |
| curve | arrow curvature. |
| lwd | line width of arrow. |
| arr.pos | Relative position of arrowheads on arrows. |
| arr.width | arrow width. |
| arr.lwd | line width of arrow, connecting two different points, (one value, or a matrix with same dimensions as x). |
| cex.txt | Relative size of text on arrows. |
| show.values | should values be shown when x is a matrix. |
| ... | Further arguments passed on to plotmat. |

## Note

innerplot uses the function plotmat in package diagram.
https://cran.r-project.org/package=diagram/vignettes/diagram.pdf

## See Also

plot.plspm, outerplot

---

it.reb *Iterative steps of Response-Based Unit Segmentation (REBUS)*

---

## Description

REBUS-PLS is an iterative algorithm for performing response based clustering in a PLS-PM framework. `it.reb` allows to perform the iterative steps of the REBUS-PLS Algorithm. It provides summarized results for final local models and the final partition of the units. Before running this function, it is necessary to run the `res.clus` function to choose the number of classes to take into account.

## Usage

```
it.reb(pls, hclus.res, nk, Y = NULL, stop.crit = 0.005,
  iter.max = 100)
```

## Arguments

| | |
|---|---|
| `pls` | an object of class "plspm" |
| `hclus.res` | object of class "res.clus" returned by `res.clus` |
| `nk` | integer larger than 1 indicating the number of classes. This value should be defined according to the dendrogram obtained by performing `res.clus`. |
| `Y` | optional data matrix used when `pls$data` is `NULL` |
| `stop.crit` | Number indicating the stop criterion for the iterative algorithm. It is suggested to use the threshold of less than 0.05% of units changing class from one iteration to the other as stopping rule. |
| `iter.max` | integer indicating the maximum number of iterations |

## Value

an object of class "rebus"

| | |
|---|---|
| `loadings` | Matrix of standardized loadings (i.e. correlations with LVs.) for each local model |
| `path.coefs` | Matrix of path coefficients for each local model |
| `quality` | Matrix containing the average communalities, the average redundancies, the R2 values, and the GoF index for each local model |
| `segments` | Vector defining the class membership of each unit |
| `origdata.clas` | The numeric matrix with original data and with a new column defining class membership of each unit |

## Author(s)

Laura Trinchera, Gaston Sanchez

## References

Esposito Vinzi, V., Trinchera, L., Squillacciotti, S., and Tenenhaus, M. (2008) REBUS-PLS: A Response-Based Procedure for detecting Unit Segments in PLS Path Modeling. *Applied Stochastic Models in Business and Industry (ASMBI)*, **24**, pp. 439-458.

Trinchera, L. (2007) Unobserved Heterogeneity in Structural Equation Models: a new approach to latent class detection in PLS Path Modeling. *Ph.D. Thesis*, University of Naples "Federico II", Naples, Italy.

## See Also

plspm, rebus.pls, res.clus

## Examples

```
## Not run:
## Example of REBUS PLS with simulated data

# load simdata
data("simdata", package='plspm')

# Calculate global plspm
sim_inner = matrix(c(0,0,0,0,0,0,1,1,0), 3, 3, byrow=TRUE)
dimnames(sim_inner) = list(c("Price", "Quality", "Satisfaction"),
                           c("Price", "Quality", "Satisfaction"))
sim_outer = list(c(1,2,3,4,5), c(6,7,8,9,10), c(11,12,13))
sim_mod = c("A", "A", "A")  # reflective indicators
sim_global = plspm(simdata, sim_inner,
                   sim_outer, modes=sim_mod)
sim_global

## Then compute cluster analysis on residuals of global model
sim_clus = res.clus(sim_global)

## To complete REBUS, run iterative algorithm
rebus_sim = it.reb(sim_global, sim_clus, nk=2,
                   stop.crit=0.005, iter.max=100)

## You can also compute complete outputs
## for local models by running:
local_rebus = local.models(sim_global, rebus_sim)

# Display plspm summary for first local model
summary(local_rebus$loc.model.1)

## End(Not run)
```

local.models *PLS-PM for global and local models*

## Description

Calculates PLS-PM for global and local models from a given partition

## Usage

```
local.models(pls, y, Y = NULL)
```

## Arguments

| | |
|---|---|
| pls | An object of class "plspm" |
| y | One object of the following classes: "rebus", "integer", or "factor", that provides the class partitions. |
| Y | Optional dataset (matrix or data frame) used when argument dataset=NULL inside pls. |

## Details

local.models calculates PLS-PM for the global model (i.e. over all observations) as well as PLS-PM for local models (i.e. observations of different partitions).

When y is an object of class "rebus", local.models is applied to the classes obtained from the REBUS algorithm.

When y is an integer vector or a factor, the values or levels are assumed to represent the group to which each observation belongs. In this case, the function local.models calculates PLS-PM for the global model, as well as PLS-PM for each group (local models).

When the object pls does not contain a data matrix (i.e. pls$data=NULL), the user must provide the data matrix or data frame in Y.

The original parameters modes, scheme, scaled, tol, and iter from the object pls are taken.

## Value

An object of class "local.models", basically a list of length k+1, where k is the number of classes.

| | |
|---|---|
| glob.model | PLS-PM of the global model |
| loc.model.1 | PLS-PM of segment (class) 1 |
| loc.model.2 | PLS-PM of segment (class) 2 |
| loc.model.k | PLS-PM of segment (class) k |

## Note

Each element of the list is an object of class "plspm". Thus, in order to examine the results for each local model, it is necessary to use the summary function.

### Author(s)

Laura Trinchera, Gaston Sanchez

### See Also

rebus.pls

### Examples

```
## Not run:
## Example of REBUS PLS with simulated data

# load simdata
data("simdata", package='plspm')

# Calculate global plspm
sim_inner = matrix(c(0,0,0,0,0,0,1,1,0), 3, 3, byrow=TRUE)
dimnames(sim_inner) = list(c("Price", "Quality", "Satisfaction"),
                           c("Price", "Quality", "Satisfaction"))
sim_outer = list(c(1,2,3,4,5), c(6,7,8,9,10), c(11,12,13))
sim_mod = c("A", "A", "A")  # reflective indicators
sim_global = plspm(simdata, sim_inner,
                   sim_outer, modes=sim_mod)
sim_global

## Then compute cluster analysis on residuals of global model
sim_clus = res.clus(sim_global)

## To complete REBUS, run iterative algorithm
rebus_sim = it.reb(sim_global, sim_clus, nk=2,
                   stop.crit=0.005, iter.max=100)

## You can also compute complete outputs
## for local models by running:
local_rebus = local.models(sim_global, rebus_sim)

# Display plspm summary for first local model
summary(local_rebus$loc.model.1)

## End(Not run)
```

---

## mobile — *ECSI Mobile Phone Provider dataset*

---

### Description

This table contains data from the article by Tenenhaus *et al.* (2005), see reference below.

## Usage

```
data(mobile)
```

## Format

A data frame with 250 observations on 24 variables on a scale from 0 to 100. The variables refer to seven latent concepts: 1) `IMAG`=Image, 2) `EXPE`=Expectations, 3) `QUAL`=Quality, 4) `VAL`=Value, 5) `SAT`=Satisfaction, 6) `COM`=Complaints, and 7) `LOY`=Loyalty.

`IMAG`: Includes variables such as reputation, trustworthiness, seriousness, and caring about customer's needs.
`EXPE`: Includes variables such as products and services provided and expectations for the overall quality.
`QUAL`: Includes variables such as reliable products and services, range of products and services, and overall perceived quality.
`VAL`: Includes variables such as quality relative to price, and price relative to quality.
`SAT`: Includes variables such as overall rating of satisfaction, fulfillment of expectations, satisfaction relative to other phone providers.
`COM`: Includes one variable defining how well or poorly custmer's complaints were handled.
`LOY`: Includes variables such as propensity to choose the same phone provider again, propensity to switch to other phone provider, intention to recommend the phone provider to friends.

`ima1` First MV of the block Image

`ima2` Second MV of the block Image

`ima3` Third MV of the block Image

`ima4` Fourth MV of the block Image

`ima5` Fifth MV of the block Image

`exp1` First MV of the block Expectations

`exp2` Second MV of the block Expectations

`exp3` Third MV of the block Expectations

`qua1` First MV of the block Quality

`qua2` Second MV of the block Quality

`qua3` Third MV of the block Quality

`qua4` Fourth MV of the block Quality

`qua5` Fifth MV of the block Quality

`qua6` Sixth MV of the block Quality

`qua7` Seventh MV of the block Quality

`val1` First MV of the block Value

`val2` Second MV of the block Value

`sat1` First MV of the block Satisfaction

`sat2` Second MV of the block Satisfaction

`sat3` Third MV of the block Satisfaction `comp` First MV of the block Complaints

`loy1` First MV of the block Loyalty

`loy2` Second MV of the block Loyalty

`loy3` Third MV of the block Loyalty

### References

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

### Examples

```
data(mobile)
```

---

## `offense` *Offense dataset*

---

### Description

Dataset with offense statistics of American football teams from the NFL (2010-2011 season).

### Usage

```
data(offense)
```

### Format

A data frame with 32 teams on the following 17 variables. The variables may be used to construct five suggested latent concepts: 1) `RUSH`=Rushing Quality, 2) `PASS`=Passing Quality, 3) `SPEC`=Special Teams and Other, 4) `SCORING`=Scoring Success, 5)`OFFENSE`=Offense Performance

| *Num* | *Variable* | *Description* | *Concept* |
|---|---|---|---|
| 1 | `YardsRushAtt` | Yards per Rush Attempt | `RUSH` |
| 2 | `RushYards` | Rush Yards per game | `RUSH` |
| 3 | `RushFirstDown` | Rush First Downs per game | `RUSH` |
| 4 | `YardsPassComp` | Yards Pass Completion | `PASS` |
| 5 | `PassYards` | Passed Yards per game | `PASS` |
| 6 | `PassFirstDown` | Pass First Downs per game | `PASS` |
| 7 | `FieldGoals` | Field Goals per game | `SPEC` |
| 8 | `OtherTDs` | Other Touchdowns (non-offense) per game | `SPEC` |
| 9 | `PointsGame` | Points per game | `SCORING` |
| 10 | `OffensTD` | Offense Touchdowns per game | `SCORING` |
| 11 | `TDGame` | Touchdowns per game | `SCORING` |
| 12 | `PassTDG` | Passing Touchdowns per game | `OFFENSE` |
| 13 | `RushTDG` | Rushing Touchdowns per game | `OFFENSE` |
| 14 | `PlaysGame` | Plays per game | `OFFENSE` |

| | | | |
|---|---|---|---|
| 15 | `YardsPlay` | Yards per Play | `OFFENSE` |
| 16 | `FirstDownPlay` | First Downs per Play | `OFFENSE` |
| 17 | `OffTimePossPerc` | Offense Time Possession Percentage | `OFFENSE` |

### Source

https://www.teamrankings.com/nfl/stats/

### Examples

```
# load data
data(offense)

# take a peek
head(offense)
```

---

## `orange` *Orange Juice dataset*

---

### Description

This data set contains the physico-chemical, sensory and hedonic measurements of 6 orange juices.

### Format

A data frame with 6 observations and 112 variables. The variables refer to three latent concepts: 1) `PHYCHEM`=Physico-Chemical, 2) `SENSORY`=Sensory, and 3) `HEDONIC`=Hedonic.

| *Num* | *Variable* | *Description* | *Concept* |
|---|---|---|---|
| 1 | glucose | Glucose (g/l) | physico-chemical |
| 2 | fructose | Fructose (g/l) | physico-chemical |
| 3 | saccharose | Saccharose (g/l) | physico-chemical |
| 4 | sweet.power | Sweetening power (g/l) | physico-chemical |
| 5 | ph1 | pH before processing | physico-chemical |
| 6 | ph2 | pH after centrifugation | physico-chemical |
| 7 | titre | Titre (meq/l) | physico-chemical |
| 8 | citric.acid | Citric acid (g/l) | physico-chemical |
| 9 | vitamin.c | Vitamin C (mg/100g) | physico-chemical |
| 10 | smell.int | Smell intensity | sensory |
| 11 | odor.typi | Odor typicity | sensory |
| 12 | pulp | Pulp | sensory |
| 13 | taste.int | Taste intensity | sensory |
| 14 | acidity | Acidity | sensory |
| 15 | bitter | Bitterness | sensory |
| 16 | sweet | Sweetness | sensory |
| 17 | judge1 | Ratings of judge 1 | hedonic |

| | | | |
|---|---|---|---|
| 18 | judge2 | Ratings of judge 2 | hedonic |
| ... | ... | ... | ... |
| 112 | judge96 | Ratings of judge 96 | hedonic |

### Source

Laboratoire de Mathematiques Appliques, Agrocampus, Rennes.

### References

Tenenhaus, M., Pages, J., Ambroisine, L., and Guinot, C. (2005) PLS methodology to study relationships between hedonic jedgements and product characteristics. *Food Quality and Preference*, **16**(4), pp. 315-325.

Pages, J., and Tenenhaus, M. (2001) Multiple factor analysis combined with PLS path modelling. Application to the analysis of relationships between physicochemical, sensory profiles and hedonic judgements. *Chemometrics and Intelligent Laboratory Systems*, **58**, pp. 261-273.

Pages, J. (2004) Multiple Factor Analysis: Main Features and Application to Sensory Data. *Revista Colombiana de Estadistica*, **27**, pp. 1-26.

### Examples

```
data(orange)
orange
```

---

## outerplot *Plot outer model*

---

### Description

Plot either outer weights or loadings in the outer model for objects of class "plspm"

### Usage

```
outerplot(x, what = "loadings", colpos = "#6890c4BB",
  colneg = "#f9675dBB", box.prop = 0.55, box.size = 0.08,
  box.cex = 1, box.col = "gray95", lcol = "gray95",
  box.lwd = 2, txt.col = "gray40", shadow.size = 0,
  curve = 0, lwd = 2, arr.pos = 0.5, arr.width = 0.15,
  cex.txt = 0.9, ...)
```

### Arguments

| | |
|---|---|
| `x` | An object of class "plspm". |
| `what` | What to plot: "loadings" or "weights". |
| `colpos` | Color of arrows for positive path coefficients. |
| `colneg` | Color of arrows for negative path coefficients. |

| | |
|---|---|
| box.prop | Length/width ratio of ellipses and rectangles. |
| box.size | Size of ellipses and rectangles. |
| box.cex | Relative size of text in ellipses and rectangles. |
| box.col | fill color of ellipses and rectangles. |
| lcol | border color of ellipses and rectangles. |
| box.lwd | line width of the box. |
| txt.col | color of text in ellipses and rectangles. |
| shadow.size | Relative size of shadow of label box. |
| curve | arrow curvature. |
| lwd | line width of arrow. |
| arr.pos | Relative position of arrowheads on arrows. |
| arr.width | arrow width. |
| cex.txt | Relative size of text on arrows. |
| ... | Further arguments passed on to plotmat. |

## Note

outerplot uses the function plotmat of package diagram.
https://cran.r-project.org/package=diagram/vignettes/diagram.pdf

## See Also

innerplot, plot.plspm, plspm

---

plot.plspm *Plots for PLS Path Models*

---

## Description

Plot method for objects of class "plspm". This function plots either the inner (i.e. structural) model with the estimated path coefficients, or the outer (i.e. measurement) model with loadings or weights.

## Usage

```
## S3 method for class 'plspm'
plot(x, what = "inner",
   colpos = "#6890c4BB", colneg = "#f9675dBB",
   box.prop = 0.55, box.size = 0.08, box.cex = 1,
   box.col = "gray95", lcol = "gray95",
   txt.col = "gray40", arr.pos = 0.5, cex.txt = 0.9, ...)
```

## Arguments

| | |
|---|---|
| x | An object of class "plspm". |
| what | What to plot: "inner", "loadings", "weights". |
| colpos | Color of arrows for positive path coefficients. |
| colneg | Color of arrows for negative path coefficients. |
| box.prop | Length/width ratio of ellipses and rectangles. |
| box.size | Size of ellipses and rectangles. |
| box.cex | Relative size of text in ellipses and rectangles. |
| box.col | fill color of ellipses and rectangles. |
| lcol | border color of ellipses and rectangles. |
| txt.col | color of text in ellipses and rectangles. |
| arr.pos | Relative position of arrowheads on arrows. |
| cex.txt | Relative size of text on arrows. |
| ... | Further arguments passed on to plotmat. |

## Details

plot.plspm is just a wraper of innerplot and outerplot.

## Note

Function plot.plspm is based on the function plotmat of package diagram.
https://cran.r-project.org/package=diagram/vignettes/diagram.pdf

## See Also

innerplot, outerplot, plspm

## Examples

```
## Not run:
 ## typical example of PLS-PM in customer satisfaction analysis
 ## model with six LVs and reflective indicators
 # load data satisfaction
 data(satisfaction)

 # define inner model matrix
 IMAG = c(0,0,0,0,0,0)
 EXPE = c(1,0,0,0,0,0)
 QUAL = c(0,1,0,0,0,0)
 VAL = c(0,1,1,0,0,0)
 SAT = c(1,1,1,1,0,0)
 LOY = c(1,0,0,0,1,0)
 sat.inner = rbind(IMAG, EXPE, QUAL, VAL, SAT, LOY)

 # define outer model list
```

```
sat.outer = list(1:5, 6:10, 11:15, 16:19, 20:23, 24:27)

# define vector of reflective modes
sat.mod = rep("A", 6)

# apply plspm
satpls = plspm(satisfaction, sat.inner, sat.outer, sat.mod, scheme="centroid",
              scaled=FALSE)

# plot path coefficients
plot(satpls, what="inner")

# plot loadings
plot(satpls, what="loadings")

# plot outer weights
plot(satpls, what="weights")

## End(Not run)
```

---

plspm *PLS-PM: Partial Least Squares Path Modeling*

---

## Description

Estimate path models with latent variables by partial least squares approach (for both metric and non-metric data)

Estimate path models with latent variables by partial least squares approach (for both metric and non-metric data)

## Usage

```
plspm(Data, path_matrix, blocks, modes = NULL,
  scaling = NULL, scheme = "centroid", scaled = TRUE,
  tol = 1e-06, maxiter = 100, plscomp = NULL,
  boot.val = FALSE, br = NULL, dataset = TRUE)
```

## Arguments

| | |
|---|---|
| Data | matrix or data frame containing the manifest variables. |
| path_matrix | A square (lower triangular) boolean matrix representing the inner model (i.e. the path relationships between latent variables). |
| blocks | list of vectors with column indices or column names from Data indicating the sets of manifest variables forming each block (i.e. which manifest variables correspond to each block). |

| | |
|---|---|
| scaling | optional argument for runing the non-metric approach; it is a list of string vectors indicating the type of measurement scale for each manifest variable specified in blocks. scaling must be specified when working with non-metric variables. Possible values: "num" (linear transformation, suitable for numerical variables), "raw" (no transformation), "nom" (non-monotonic transformation, suitable for nominal variables), and "ord" (monotonic transformation, suitable for ordinal variables). |
| modes | character vector indicating the type of measurement for each block. Possible values are: "A", "B", "newA", "PLScore", "PLScow". The length of modes must be equal to the length of blocks. |
| scheme | string indicating the type of inner weighting scheme. Possible values are "centroid", "factorial", or "path". |
| scaled | whether manifest variables should be standardized. Only used when scaling = NULL. When (TRUE, data is scaled to standardized values (mean=0 and variance=1). The variance is calculated dividing by N instead of N-1). |
| tol | decimal value indicating the tolerance criterion for the iterations (tol=0.000001). Can be specified between 0 and 0.001. |
| maxiter | integer indicating the maximum number of iterations (maxiter=100 by default). The minimum value of maxiter is 100. |
| plscomp | optional vector indicating the number of PLS components (for each block) to be used when handling non-metric data (only used if scaling is provided) |
| boot.val | whether bootstrap validation should be performed. (FALSE by default). |
| br | number bootstrap resamples. Used only when boot.val=TRUE. When boot.val=TRUE, the default number of re-samples is 100. |
| dataset | whether the data matrix used in the computations should be retrieved (TRUE by default). |

## Details

The function plspm estimates a path model by partial least squares approach providing the full set of results.

The argument path_matrix is a matrix of zeros and ones that indicates the structural relationships between latent variables. path_matrix must be a lower triangular matrix; it contains a 1 when column j affects row i, 0 otherwise.

- plspm: Partial Least Squares Path Modeling
- plspm.fit: Simple version for PLS-PM
- plspm.groups: Two Groups Comparison in PLS-PM
- rebus.pls: Response Based Unit Segmentation (REBUS)

## Value

An object of class "plspm".

| | |
|---|---|
| outer_model | Results of the outer model. Includes: outer weights, standardized loadings, communalities, and redundancies |
| inner_model | Results of the inner (structural) model. Includes: path coeffs and R-squared for each endogenous latent variable |
| scores | Matrix of latent variables used to estimate the inner model. If scaled=FALSE then scores are latent variables calculated with the original data (non-stardardized). |
| path_coefs | Matrix of path coefficients (this matrix has a similar form as path_matrix) |
| crossloadings | Correlations between the latent variables and the manifest variables (also called crossloadings) |
| inner_summary | Summarized results of the inner model. Includes: type of LV, type of measurement, number of indicators, R-squared, average communality, average redundancy, and average variance extracted |
| effects | Path effects of the structural relationships. Includes: direct, indirect, and total effects |
| unidim | Results for checking the unidimensionality of blocks (These results are only meaningful for reflective blocks) |
| gof | Goodness-of-Fit index |
| data | Data matrix containing the manifest variables used in the model. Only available when dataset=TRUE |
| boot | List of bootstrapping results; only available when argument boot.val=TRUE |

## Author(s)

Gaston Sanchez, Giorgio Russolillo

## References

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

Lohmoller J.-B. (1989) *Latent variables path modeling with partial least squares.* Heidelberg: Physica-Verlag.

Wold H. (1985) Partial Least Squares. In: Kotz, S., Johnson, N.L. (Eds.), *Encyclopedia of Statistical Sciences*, Vol. 6. Wiley, New York, pp. 581-591.

Wold H. (1982) Soft modeling: the basic design and some extensions. In: K.G. Joreskog & H. Wold (Eds.), *Systems under indirect observations: Causality, structure, prediction*, Part 2, pp. 1-54. Amsterdam: Holland.

Russolillo, G. (2012) Non-Metric Partial Least Squares. *Electronic Journal of Statistics*, **6**, pp. 1641-1669. https://projecteuclid.org/euclid.ejs/1348665231

## See Also

innerplot, outerplot,

## Examples

```
## Not run:
## typical example of PLS-PM in customer satisfaction analysis
## model with six LVs and reflective indicators

# load dataset satisfaction
data(satisfaction)

# path matrix
IMAG = c(0,0,0,0,0,0)
EXPE = c(1,0,0,0,0,0)
QUAL = c(0,1,0,0,0,0)
VAL = c(0,1,1,0,0,0)
SAT = c(1,1,1,1,0,0)
LOY = c(1,0,0,0,1,0)
sat_path = rbind(IMAG, EXPE, QUAL, VAL, SAT, LOY)

# plot diagram of path matrix
innerplot(sat_path)

# blocks of outer model
sat_blocks = list(1:5, 6:10, 11:15, 16:19, 20:23, 24:27)

# vector of modes (reflective indicators)
sat_mod = rep("A", 6)

# apply plspm
satpls = plspm(satisfaction, sat_path, sat_blocks, modes = sat_mod,
   scaled = FALSE)

# plot diagram of the inner model
innerplot(satpls)

# plot loadings
outerplot(satpls, what = "loadings")

# plot outer weights
outerplot(satpls, what = "weights")

## End(Not run)
```

---

`plspm.fit` *Basic results for Partial Least Squares Path Modeling*

---

## Description

Estimate path models with latent variables by partial least squares approach without providing the full list of results as `plspm()`. This might be helpful when doing simulations, intensive computations, or when you don't want the whole enchilada.

## Usage

```
plspm.fit(Data, path_matrix, blocks, modes = NULL,
  scaling = NULL, scheme = "centroid", scaled = TRUE,
  tol = 1e-06, maxiter = 100, plscomp = NULL)
```

## Arguments

| | |
|---|---|
| Data | matrix or data frame containing the manifest variables. |
| path_matrix | A square (lower triangular) boolean matrix representing the inner model (i.e. the path relationships betwenn latent variables). |
| blocks | list of vectors with column indices or column names from Data indicating the sets of manifest variables forming each block (i.e. which manifest variables correspond to each block). |
| scaling | optional list of string vectors indicating the type of measurement scale for each manifest variable specified in blocks. scaling must be specified when working with non-metric variables. |
| modes | character vector indicating the type of measurement for each block. Possible values are: "A", "B", "newA", "PLScore", "PLScow". The length of modes must be equal to the length of blocks. |
| scheme | string indicating the type of inner weighting scheme. Possible values are "centroid", "factorial", or "path". |
| scaled | whether manifest variables should be standardized. Only used when scaling = NULL. When (TRUE data is scaled to standardized values (mean=0 and variance=1). The variance is calculated dividing by N instead of N-1). |
| tol | decimal value indicating the tolerance criterion for the iterations (tol=0.000001). Can be specified between 0 and 0.001. |
| maxiter | integer indicating the maximum number of iterations (maxiter=100 by default). The minimum value of maxiter is 100. |
| plscomp | optional vector indicating the number of PLS components (for each block) to be used when handling non-metric data (only used if scaling is provided) |

## Details

plspm.fit performs the basic PLS algorithm and provides limited results (e.g. outer model, inner model, scores, and path coefficients).

The argument path_matrix is a matrix of zeros and ones that indicates the structural relationships between latent variables. path_matrix must be a lower triangular matrix; it contains a 1 when column j affects row i, 0 otherwise.

## Value

An object of class "plspm".

| | |
|---|---|
| outer_model | Results of the outer model. Includes: outer weights, standardized loadings, communalities, and redundancies |
| inner_model | Results of the inner (structural) model. Includes: path coeffs and R-squared for each endogenous latent variable |
| scores | Matrix of latent variables used to estimate the inner model. If scaled=FALSE then scores are latent variables calculated with the original data (non-stardardized). If scaled=TRUE then scores and latents have the same values |
| path_coefs | Matrix of path coefficients (this matrix has a similar form as path_matrix) |

## Author(s)

Gaston Sanchez, Giorgio Russolillo

## References

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

Lohmoller J.-B. (1989) *Latent variables path modeling with partial least squares.* Heidelberg: Physica-Verlag.

Wold H. (1985) Partial Least Squares. In: Kotz, S., Johnson, N.L. (Eds.), *Encyclopedia of Statistical Sciences*, Vol. 6. Wiley, New York, pp. 581-591.

Wold H. (1982) Soft modeling: the basic design and some extensions. In: K.G. Joreskog & H. Wold (Eds.), *Systems under indirect observations: Causality, structure, prediction*, Part 2, pp. 1-54. Amsterdam: Holland.

## See Also

innerplot, plot.plspm,

## Examples

```
## Not run:
 ## typical example of PLS-PM in customer satisfaction analysis
 ## model with six LVs and reflective indicators

 # load dataset satisfaction
 data(satisfaction)

 # inner model matrix
 IMAG = c(0,0,0,0,0,0)
 EXPE = c(1,0,0,0,0,0)
 QUAL = c(0,1,0,0,0,0)
 VAL = c(0,1,1,0,0,0)
 SAT = c(1,1,1,1,0,0)
 LOY = c(1,0,0,0,1,0)
 sat_path = rbind(IMAG, EXPE, QUAL, VAL, SAT, LOY)

 # outer model list
 sat_blocks = list(1:5, 6:10, 11:15, 16:19, 20:23, 24:27)
```

```
# vector of reflective modes
sat_modes = rep("A", 6)

# apply plspm.fit
satpls = plspm.fit(satisfaction, sat_path, sat_blocks, sat_modes,
    scaled=FALSE)

# summary of results
summary(satpls)

# default plot (inner model)
plot(satpls)

## End(Not run)
```

---

`plspm.groups` *Two Groups Comparison in PLS-PM*

---

### Description

Performs a group comparison test for comparing path coefficients between two groups. The null and alternative hypotheses to be tested are: H0: path coefficients are not significantly different; H1: path coefficients are significantly different

### Usage

```
plspm.groups(pls, group, Y = NULL, method = "bootstrap",
  reps = NULL)
```

### Arguments

| | |
|---|---|
| `pls` | object of class `"plspm"` |
| `group` | factor with 2 levels indicating the groups to be compared |
| `Y` | optional dataset (matrix or data frame) used when argument `dataset=NULL` inside `pls`. |
| `method` | method to be used in the test. Possible values are `"bootstrap"` or `"permutation"` |
| `reps` | integer indicating the number of either bootstrap resamples or number of permutations. If `NULL` then `reps`=100 |

### Details

`plspm.groups` performs a two groups comparison test in PLS-PM for comparing path coefficients between two groups. Only two methods are available: 1) bootstrap, and 2) permutation. The bootstrap test is an adapted t-test based on bootstrap standard errors. The permutation test is a randomization test which provides a non-parametric option.

When the object `pls` does not contain a data matrix (i.e. `pls$data=NULL`), the user must provide the data matrix or data frame in `Y`.

### Value

An object of class "plspm.groups"

- `test` Table with the results of the applied test. Includes: path coefficients of the global model, path coeffs of group1, path coeffs of group2, (absolute) difference of path coeffs between groups, and the test results with the p-value.
- `global` List with inner model results for the global model
- `group1` List with inner model results for group1
- `group2` List with inner model results for group2

### Author(s)

Gaston Sanchez

### References

Chin, W.W. (2003) A permutation procedure for multi-group comparison of PLS models. In: Vilares M., Tenenhaus M., Coelho P., Esposito Vinzi V., Morineau A. (Eds.) *PLS and Related Methods - Proceedings of the International Symposium PLS03.* Decisia, pp. 33-43.

Chin, W.W. (2000) Frequently Asked Questions, Partial Least Squares PLS-Graph.

### See Also

plspm

### Examples

```
## Not run:
 ## example with customer satisfaction analysis
 ## group comparison based on the segmentation variable "gender"

 # load data satisfaction
 data(satisfaction)

 # define inner model matrix
 IMAG = c(0,0,0,0,0,0)
 EXPE = c(1,0,0,0,0,0)
 QUAL = c(0,1,0,0,0,0)
 VAL = c(0,1,1,0,0,0)
 SAT = c(1,1,1,1,0,0)
 LOY = c(1,0,0,0,1,0)
 sat_path = rbind(IMAG, EXPE, QUAL, VAL, SAT, LOY)

 # define outer model list
 sat_blocks = list(1:5, 6:10, 11:15, 16:19, 20:23, 24:27)

 # define vector of reflective modes
 sat_mod = rep("A", 6)

 # apply plspm
```

```
 satpls = plspm(satisfaction, sat_path, sat_blocks,
               modes = sat_mod, scaled = FALSE)

 # permutation test with 100 permutations
 group_perm = plspm.groups(satpls, satisfaction$gender,
                          method="permutation", reps=100)
 group_perm

## End(Not run)
```

---

quantiplot *Quantification Plot*

---

### Description

Quantification Plots for Non-Metric PLS-PM

### Usage

```
quantiplot(pls, lv = NULL, mv = NULL, pch = 16,
  col = "darkblue", lty = 2, ...)
```

### Arguments

| | |
|---|---|
| pls | a non-metric "plspm" object |
| lv | number or name of latent variable |
| mv | number or name of manifest variable |
| pch | Either an integer specifying a symbol or a single character to be used as the default in plotting points |
| col | color |
| lty | type of line |
| ... | Further arguments passed on to plot. |

### Details

If both lv and mv are specified, only the value of lv will be taken into account.
If the given lv have more than 15 variables, only the first 15 are plotted.

`rebus.pls` *Response Based Unit Segmentation (REBUS)*

## Description

Performs all the steps of the REBUS-PLS algorithm. Starting from the global model, REBUS allows us to detect local models with better performance.

## Usage

```
rebus.pls(pls, Y = NULL, stop.crit = 0.005,
  iter.max = 100)
```

## Arguments

| | |
|---|---|
| `pls` | Object of class `"plspm"` |
| `Y` | Optional dataset (matrix or data frame) used when argument `dataset=NULL` inside `pls`. |
| `stop.crit` | Number indicating the stop criterion for the iterative algorithm. Use a threshold of less than 0.05% of units changing class from one iteration to the other as stopping rule. |
| `iter.max` | integer indicating the maximum number of iterations. |

## Value

An object of class `"rebus"`, basically a list with:

| | |
|---|---|
| `loadings` | Matrix of standardized loadings (i.e. correlations with LVs.) for each local model. |
| `path.coefs` | Matrix of path coefficients for each local model. |
| `quality` | Matrix containing the average communalities, average redundancies, R2 values, and GoF values for each local model. |
| `segments` | Vector defining for each unit the class membership. |
| `origdata.clas` | The numeric matrix with original data and with a new column defining class membership of each unit. |

## Author(s)

Laura Trinchera, Gaston Sanchez

### References

Esposito Vinzi V., Trinchera L., Squillacciotti S., and Tenenhaus M. (2008) REBUS-PLS: A Response-Based Procedure for detecting Unit Segments in PLS Path Modeling. *Applied Stochastic Models in Business and Industry (ASMBI)*, **24**, pp. 439-458.

Trinchera, L. (2007) Unobserved Heterogeneity in Structural Equation Models: a new approach to latent class detection in PLS Path Modeling. *Ph.D. Thesis*, University of Naples "Federico II", Naples, Italy.

### See Also

plspm, res.clus, it.reb, rebus.test, local.models

### Examples

```
## Not run:
 ## typical example of PLS-PM in customer satisfaction analysis
 ## model with six LVs and reflective indicators
 ## example of rebus analysis with simulated data

 # load data
 data(simdata)

 # Calculate plspm
 sim_inner = matrix(c(0,0,0,0,0,0,1,1,0), 3, 3, byrow=TRUE)
 dimnames(sim_inner) = list(c("Price", "Quality", "Satisfaction"),
                            c("Price", "Quality", "Satisfaction"))
 sim_outer = list(c(1,2,3,4,5), c(6,7,8,9,10), c(11,12,13))
 sim_mod = c("A", "A", "A")  # reflective indicators
 sim_global = plspm(simdata, sim_inner,
                    sim_outer, modes=sim_mod)
 sim_global

 # run rebus.pls and choose the number of classes
 # to be taken into account according to the displayed dendrogram.
 rebus_sim = rebus.pls(sim_global, stop.crit = 0.005, iter.max = 100)

 # You can also compute complete outputs for local models by running:
 local_rebus = local.models(sim_global, rebus_sim)

## End(Not run)
```

---

`rebus.test` *Permutation Test for REBUS Multi-Group Comparison*

---

### Description

Performs permutation tests for comparing pairs of groups from a REBUS object.

### Usage

```
rebus.test(pls, reb, Y = NULL)
```

### Arguments

| | |
|---|---|
| pls | Object of class "plspm" returned by plspm |
| reb | Object of class "rebus" returned by either rebus.pls or it.reb. |
| Y | Optional dataset (matrix or data frame) used when argument dataset=NULL inside pls. |

### Details

A permutation test on path coefficients, loadings, and GoF index is applied to the classes obtained from REBUS, by comparing two classes at a time. That is to say, a permutation test is applied on pair of classes. The number of permutations in each test is 100. In turn, the number of classes handled by rebus.test is limited to 6.

When pls$data=NULL (there is no data matrix), the user must provide the data matrix or data frame in Y.

### Value

An object of class "rebus.test", basically a list containing the results of each pair of compared classes. In turn, each element of the list is also a list with the results for the path coefficients, loadings, and GoF index.

### Author(s)

Laura Trinchera, Gaston Sanchez

### References

Chin, W.W. (2003) A permutation procedure for multi-group comparison of PLS models. In: Vilares M., Tenenhaus M., Coelho P., Esposito Vinzi V., Morineau A. (Eds.) *PLS and Related Methods - Proceedings of the International Symposium PLS03.* Decisia, pp. 33-43.

### See Also

rebus.pls, local.models

### Examples

```
## Not run:
 ## typical example of PLS-PM in customer satisfaction analysis
 ## model with six LVs and reflective indicators
 ## example of rebus analysis with simulated data

 # load data
 data(simdata)
```

```
# Calculate plspm
sim_path = matrix(c(0,0,0,0,0,0,1,1,0), 3, 3, byrow=TRUE)
dimnames(sim_path) = list(c("Price", "Quality", "Satisfaction"),
                          c("Price", "Quality", "Satisfaction"))
sim_blocks = list(c(1,2,3,4,5), c(6,7,8,9,10), c(11,12,13))
sim_mod = c("A", "A", "A")  # reflective indicators
sim_global = plspm(simdata, sim_path,
                   sim_blocks, modes=sim_mod)
sim_global

# Cluster analysis on residuals of global model
sim_clus = res.clus(sim_global)

# Iterative steps of REBUS algorithm on 2 classes
rebus_sim = it.reb(sim_global, sim_clus, nk=2,
                   stop.crit=0.005, iter.max=100)

# apply rebus.test
sim_permu = rebus.test(sim_global, rebus_sim)

# inspect sim.rebus
sim_permu
sim_permu$test_1_2

# or equivalently
sim_permu[[1]]

## End(Not run)
```

---

`res.clus` *Clustering on communality and structural residuals*

---

## Description

Computes communality and structural residuals from a global PLS-PM model and performs a Hierarchical Cluster Analysis on these residuals according to the REBUS algorithm.

## Usage

```
res.clus(pls, Y = NULL)
```

## Arguments

| | |
|---|---|
| `pls` | Object of class `"plspm"` |
| `Y` | Optional dataset (matrix or data frame) used when argument `dataset=NULL` inside `pls`. |

### Details

`res.clus()` comprises the second and third steps of the REBUS-PLS Algorithm. It computes communality and structural residuals. Then it performs a Hierarchical Cluster Analysis on these residuals (step three of REBUS-PLS Algorithm). As a result, this function directly provides a dendrogram obtained from a Hierarchical Cluster Analysis.

### Value

An Object of class `"hclust"` containing the results of the Hierarchical Cluster Analysis on the communality and structural residuals.

### Author(s)

Laura Trinchera, Gaston Sanchez

### References

Esposito Vinzi V., Trinchera L., Squillacciotti S., and Tenenhaus M. (2008) REBUS-PLS: A Response-Based Procedure for detecting Unit Segments in PLS Path Modeling. *Applied Stochastic Models in Business and Industry (ASMBI)*, **24**, pp. 439-458.

Trinchera, L. (2007) Unobserved Heterogeneity in Structural Equation Models: a new approach to latent class detection in PLS Path Modeling. *Ph.D. Thesis*, University of Naples "Federico II", Naples, Italy.

### See Also

`it.reb`, `plspm`

### Examples

```
## Not run:
 ## example of rebus analysis with simulated data

 # load data
 data(simdata)

 # Calculate plspm
 sim_path = matrix(c(0,0,0,0,0,0,1,1,0), 3, 3, byrow=TRUE)
 dimnames(sim_path) = list(c("Price", "Quality", "Satisfaction"),
                           c("Price", "Quality", "Satisfaction"))
 sim_blocks = list(c(1,2,3,4,5), c(6,7,8,9,10), c(11,12,13))
 sim_modes = c("A", "A", "A")
 sim_global = plspm(simdata, sim_path,
                    sim_blocks, modes=sim_modes)
 sim_global

 # Then compute cluster analysis on the residuals of global model
 sim_clus = res.clus(sim_global)

## End(Not run)
```

`rescale` *Rescale Latent Variable Scores*

## Description

Rescale standardized latent variable scores to original scale of manifest variables

## Usage

```
rescale(pls, data = NULL)
```

## Arguments

| | |
|---|---|
| `pls` | object of class `"plspm"` |
| `data` | Optional dataset (matrix or data frame) used when argument `dataset=NULL` inside `pls`. |

## Details

`rescale` requires all outer weights to be positive

## Value

A data frame with the rescaled latent variable scores

## Author(s)

Gaston Sanchez

## See Also

`plspm`

## Examples

```
## Not run:
 ## example with customer satisfaction analysis

 # load data satisfaction
 data(satisfaction)

 # define inner model matrix
 IMAG = c(0,0,0,0,0,0)
 EXPE = c(1,0,0,0,0,0)
 QUAL = c(0,1,0,0,0,0)
 VAL = c(0,1,1,0,0,0)
 SAT = c(1,1,1,1,0,0)
 LOY = c(1,0,0,0,1,0)
 sat_path = rbind(IMAG, EXPE, QUAL, VAL, SAT, LOY)
```

```
# define outer model list
sat_blocks = list(1:5, 6:10, 11:15, 16:19, 20:23, 24:27)

# define vector of reflective modes
sat_modes = rep("A", 6)

# apply plspm
my_pls = plspm(satisfaction, sat_path, sat_blocks, modes = sat_modes,
             scaled=FALSE)

# rescaling standardized scores of latent variables
new_scores = rescale(my_pls)

# compare standardized LVs against rescaled LVs
summary(my_pls$scores)
summary(new_scores)

## End(Not run)
```

---

`rho` *Dillon-Goldstein's rho*

---

## Description

Dillon-Goldstein's rho of a single block of variables

## Usage

```
rho(X)
```

## Arguments

| | |
|---|---|
| X | matrix representing one block of manifest variables |

## Value

Dillon-Goldstein's rho

## Author(s)

Gaston Sanchez

## See Also

alpha, unidim

## Examples

```
## Not run:
 # load dataset satisfaction
 data(satisfaction)

 # block Image (first 5 columns of satisfaction)
 Image = satisfaction[,1:5]

 # compute Dillon-Goldstein's rho for Image block
 rho(Image)

## End(Not run)
```

---

`russa` *Russett A*

---

## Description

Russett dataset with variable demo as numeric variable

## Format

A data frame with 47 rows and 9 columns

---

`russb` *Russett B*

---

## Description

Russett dataset with variable demo as factor

## Format

A data frame with 47 rows and 9 columns

`russett` *Russett dataset*

## Description

Data set from Russett (1964) about agricultural inequality, industrial development and political instability.

## Usage

```
data(russett)
```

## Format

A data frame with 47 observations on the following 11 variables. The variables may be used to construct three latent concepts: 1) `AGRIN`=Agricultural Inequality, 2) `INDEV`=Industrial Development, 3) `POLINS`=Political Instability.

| *Num* | *Variable* | *Description* | *Concept* |
|---|---|---|---|
| 1 | gini | Inequality of land distribution | `AGRIN` |
| 2 | farm | Percentage of farmers that own half of the land | `AGRIN` |
| 3 | rent | Percentage of farmers that rent all their land | `AGRIN` |
| 4 | gnpr | Gross national product per capita | `INDEV` |
| 5 | labo | Percentage of labor force employed in agriculture | `INDEV` |
| 6 | inst | Instability of executive (1945-1961) | `POLINS` |
| 7 | ecks | Number of violent internal war incidents (1941-1961) | `POLINS` |
| 8 | death | Number of people killed as a result of civic group violence (1950-1962) | `POLINS` |
| 9 | demostab | Political regime: stable democracy | `POLINS` |
| 10 | demoinst | Political regime: unstable democracy | `POLINS` |
| 11 | dictator | Political regime: dictatorship | `POLINS` |

## References

Russett B.M. (1964) Inequality and Instability: The Relation of Land Tenure to Politics. *World Politics* **16:3**, pp. 442-454.

## Examples

```
data(russett)
russett
```

`satisfaction` *Satisfaction dataset*

### Description

This data set contains the variables from a customer satisfaction study of a Spanish credit institution on 250 customers.

### Format

A data frame with 250 observations and 28 variables. Variables from 1 to 27 refer to six latent concepts: 1) `IMAG`=Image, 2) `EXPE`=Expectations, 3) `QUAL`=Quality, 4) `VAL`=Value, 5) `SAT`=Satisfaction, and 6) `LOY`=Loyalty. The last variable is a categorical variable indicating the gender of the individual.

`IMAG`: Includes variables such as reputation, trustworthiness, seriousness, solidness, and caring about customer's needs.
`EXPE`: Includes variables such as products and services provided, customer service, providing solutions, and expectations for the overall quality.
`QUAL`: Includes variables such as reliable products and services, range of products and services, personal advice, and overall perceived quality.
`VAL`: Includes variables such as beneficial services and products, valuable investments, quality relative to price, and price relative to quality.
`SAT`: Includes variables such as overall rating of satisfaction, fulfillment of expectations, satisfaction relative to other banks, and performance relative to customer's ideal bank.
`LOY`: Includes variables such as propensity to choose the same bank again, propensity to switch to other bank, intention to recommend the bank to friends, and sense of loyalty.

### Source

Laboratory of Information Analysis and Modeling (LIAM). Facultat d'Informatica de Barcelona, Universitat Politecnica de Catalunya.

### Examples

```
data(satisfaction)
satisfaction
```

`simdata` *Simulated data for REBUS with two groups*

### Description

Simulated data with two latent classes showing different local models.

## Usage

```
data(simdata)
```

## Format

A data frame of simulated data with 400 observations on the following 14 variables.

`mv1` first manifest variable of the block *Price Fairness*

`mv2` second manifest variable of the block *Price Fairness*

`mv3` third manifest variable of the block *Price Fairness*

`mv4` fourth manifest variable of the block *Price Fairness*

`mv5` fifth manifest variable of the block *Price Fairness*

`mv6` first manifest variable of the block *Quality*

`mv7` second manifest variable of the block *Quality*

`mv8` third manifest variable of the block *Quality*

`mv9` fourth manifest variable of the block *Quality*

`mv10` fifth manifest variable of the block *Quality*

`mv11` first manifest variable of the block *Customer Satisfaction*

`mv12` second manifest variable of the block *Customer Satisfaction*

`mv13` third manifest variable of the block *Customer Satisfaction*

`group` a numeric vector

## Details

The postulated model overlaps the one used by Jedidi *et al.* (1997) and by Esposito Vinzi *et al.* (2007) for their numerical examples. It is composed of one latent endogenous variable, *Customer Satisfaction*, and two latent exogenous variables, *Price Fairness* and *Quality*. Each latent exogenous variable (*Price Fairness* and *Quality*) has five manifest variables (reflective mode), and the latent endogenous variable (*Customer Satisfaction*) is measured by three indicators (reflective mode).

Two latent classes showing different local models are supposed to exist. Each one is composed of 200 units. Thus, the data on the aggregate level for each one of the numerical examples includes 400 units.

The simulation scheme involves working with local models that are different at both the measurement and the structural model levels. In particular, the experimental sets of data consist of two latent classes with the following characteristics:
(a) Class 1 - price fairness seeking customers - characterized by a strong relationship between *Price Fairness* and *Customer Satisfaction* (close to 0.9) and a weak relationship between *Quality* and *Customer Satisfaction* (close to 0.1), as well as by a weak correlation between the 3rd manifest variable of the *Price Fairness block* (mv3) and the corresponding latent variable;
(b) Class 2 - quality oriented customers - characterized by a strong relationship between *Quality* and *Customer Satisfaction* (close to 0.1) and a weak relationship between *Price Fairness* and *Customer Satisfaction* (close to 0.9), as well as by a weak correlation between the 3rd manifest variable (mv8) of the *Quality* block and the corresponding latent variable.

### Source

Simulated data from Trinchera (2007). See **References** below.

### References

Esposito Vinzi, V., Ringle, C., Squillacciotti, S. and Trinchera, L. (2007) Capturing and treating unobserved heterogeneity by response based segmentation in PLS path modeling. A comparison of alternative methods by computational experiments. *Working paper*, ESSEC Business School.

Jedidi, K., Jagpal, S. and De Sarbo, W. (1997) STEMM: A general finite mixture structural equation model. *Journal of Classification* **14**, pp. 23-50.

Trinchera, L. (2007) Unobserved Heterogeneity in Structural Equation Models: a new approach to latent class detection in PLS Path Modeling. *Ph.D. Thesis*, University of Naples "Federico II", Naples, Italy.

### Examples

```
data(simdata)
simdata
```

---

## `spainfoot` *Spanish football dataset*

---

### Description

This data set contains the results of the teams in the Spanish football league 2008-2009.

### Format

A data frame with 20 observations on 14 variables. The variables may be used to construct four latent concepts: 1) `ATTACK`=Attack, 2) `DEFENSE`=Defense, 3) `SUCCESS`=Success, 4) `INDIS`=Indiscipline.

| *Num* | *Variable* | *Description* | *Concept* |
|---|---|---|---|
| 1 | GSH | Goals Scored Home: total number of goals scored at home | `ATTACK` |
| 2 | GSA | Goals Scored Away: total number of goals scored away | `ATTACK` |
| 3 | SSH | Success to Score Home: Percentage of matches with scores goals at home | `ATTACK` |
| 4 | SSA | Success to Score Away: Percentage of matches with scores goals away | `ATTACK` |
| 5 | GCH | Goals Conceded Home: total number of goals conceded at home | `DEFENSE` |
| 6 | GCA | Goals Conceded Away: total number of goals conceded away | `DEFENSE` |
| 7 | CSH | Clean Sheets Home: percentage of matches with no conceded goals at home | `DEFENSE` |
| 8 | CSA | Clean Sheets Away: percentage of matches with no conceded goals away | `DEFENSE` |
| 9 | WMH | Won Matches Home: total number of won matches at home | `SUCCESS` |
| 10 | WMA | Won matches Away: total number of won matches away | `SUCCESS` |
| 11 | LWR | Longest Winning Run: longest run of won matches | `SUCCESS` |
| 12 | LRWR | Longest Run Without Loss: longest run of matches without losing | `SUCCESS` |
| 13 | YC | Yellow Cards: total number of yellow cards | `INDIS` |
| 14 | RC | Red Cards: total number of red cards | `INDIS` |

### Source

League Day.
Cero a cero. https://www.ceroacero.es/

### Examples

```
data(spainfoot)
spainfoot
```

---

`technology` *Technology data set*

---

### Description

This data set contains the variables from a "user and acceptance of technology" model on 300 users.

### Usage

```
data(technology)
```

### Format

A data frame with 300 observations and 21 variables. Variables can be grouped in six latent concepts: 1) `PERF_EXP`=Performance Expectancy, 2) `EFF_EXP`=Effort Expectancy, 3) `SUB_NORM`=Subjective Norm, 4) `FAC_COND`=Facilitating Conditions, 5) `BEH_INT`=Behavioral Intention, and 6) `USE_BEH`=Use Behavior.

| *Num* | *Variable* | *Description* |
|---|---|---|
| 1 | pe1 | I find computers useful in my job |
| 2 | pe2 | Using computers in my job enables me to accomplish tasks more quickly |
| 3 | pe3 | Using computers in my job increases my productivity |
| 4 | pe4 | Using computers enhances my effectiveness on the job |
| 5 | ee1 | My interactions with computers are clear and understandable |
| 6 | ee2 | It is easy for me to become skillful using computers |
| 7 | ee3 | I find computers easy to use |
| 8 | ee4 | Learning to use computers is easy for me |
| 9 | sn1 | Most people who are important to me think I should use computers |
| 10 | sn2 | Most people who are important to me would want me to use computers |
| 11 | sn3 | People whose opinions I value would prefer me to use computers |
| 12 | fc1 | I have the resources and the knowledge and the ability to make use of the computer |
| 13 | fc1 | A central support was available to help with computer problems |
| 14 | fc1 | Management provided most of the necessary help and resources for computing |
| 15 | bi1 | I predict I will continue to use computers on a regular basis |

| | | |
|---|---|---|
| 16 | bi2 | What are the chances in 100 that you will continue as a computer user? |
| 17 | bi3 | To do my work, I would use computers rather than any other means available |
| 18 | use1 | On an average working day, how much time do you spend using computers? |
| 19 | use2 | On average, how frequently do you use computers? |
| 20 | use3 | How many different computer applications have you worked with or used in your job? |
| 21 | use4 | According to your job requirements, indicate each task you use computers to perform? |

## References

Venkatesh V., Morris M.G., Davis G.B., Davis F.D. (2003) User Acceptance of Information Technology: Toward a Unified View. *MIS Quarterly*, Vol. 27 (3): 425-478.

## Examples

```
data(technology)
summary(technology)
```

---

`unidim` *Unidimensionality of blocks*

---

## Description

Compute unidimensionality indices (a.k.a. Composite Reliability indices)

## Usage

```
unidim(Data, blocks = NULL)
```

## Arguments

| | |
|---|---|
| `Data` | matrix or data frame with variables |
| `blocks` | optional list with vectors indicating the variables in each block |

## Value

A data frame with the following columns:

| | |
|---|---|
| `Block` | name of block |
| `MVs` | number of manifest variables in each block |
| `C.alpha` | Cronbach's alpha |
| `DG.rho` | Dillon-Goldstein rho |
| `eig.1st` | First eigenvalue |
| `eig.2nd` | Second eigenvalue |

## Author(s)

Gaston Sanchez

## See Also

alpha, rho

## Examples

```
## Not run:
 # load dataset satisfaction
 data(satisfaction)

 # blocks Image and Expectations
 ima_expe = list(Image=1:5, Expec=6:10)

 # compute unidimensionality indices
 unidim(satisfaction, ima_expe)

## End(Not run)
```

---

wines *Wines dataset*

---

## Description

These data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines.

## Format

A data frame with 178 observations and 14 variables.

| *Num* | *Variable* | *Description* |
|---|---|---|
| 1 | class | Type of wine |
| 2 | alcohol | Alcohol |
| 3 | malic.acid | Malic acid |
| 4 | ash | Ash |
| 5 | alcalinity | Alcalinity |
| 6 | magnesium | Magnesium |
| 7 | phenols | Total phenols |
| 8 | flavanoids | Flavanoids |
| 9 | nofla.phen | Nonflavanoid phenols |
| 10 | proantho | Proanthocyanins |
| 11 | col.intens | Color intensity |
| 12 | hue | Hue |
| 13 | diluted | OD280/OD315 of diluted wines |

| | | |
|---|---|---|
| 14 | proline | Proline |

## Source

Machine Learning Repository. https://archive.ics.uci.edu/ml/datasets/Wine

## References

Forina, M. et al, PARVUS *An Extendible Package for Data Exploration, Classification and Correlation.* Institute of Pharmaceutical and Food Analysis and Technologies, Via Brigata Salerno, 16147 Genoa, Italy.

## Examples

```
data(wines)
wines
```

# Index