

Package ‘plspm’

February 15, 2012

Type Package

Title Partial Least Squares Data Analysis Methods

Version 0.2-2

Date 2012-01-22

Author Gaston Sanchez, Laura Trinchera

Maintainer Gaston Sanchez <plspm@gmail.com>

Depends amap, diagram

Suggests FactoMineR

Description Partial Least Squares (PLS) methods with emphasis on structural equation models with latent variables.

License GPL (>= 2)

URL <http://www.plsmodeling.com>

LazyLoad yes

Repository CRAN

Date/Publication 2012-01-23 07:38:47

R topics documented:

plspm-package	2
arizona	3
futbol	4
it.reb	5
local.models	7
mobile	9
nipals	11
orange	12
plot.nipals	13

plot.plsca	14
plot.plspm	15
plot.plspm.groups	16
plot.plsreg1	18
plot.plsreg2	19
plsca	20
plspm	21
plspm.groups	25
plsreg1	27
plsreg2	29
print.local.models	31
print.nipals	32
print.plsca	32
print.plspm	33
print.plspm.groups	33
print.plsreg1	34
print.plsreg2	34
print.rebus	35
print.rebus.test	36
print.summary.plspm	36
rebus.pls	37
rebus.test	39
res.clus	41
resclus.plot	43
ropes	44
russett	45
satisfaction	46
sim.data	47
spainfoot	49
summary.plspm	50
vehicles	51
wines	52

Index **53**

plspm-package	<i>Package of Partial Least Squares (PLS) data analysis methods.</i>
---------------	--

Description

Contains functions of various Partial Least Squares methods such as the NIPALS algorithm, PLS Regression (PLSR1 and PLSR2), PLS Canonical Analysis, PLS Path Modeling, and REBUS-PLS.

Details

Package: plspm
Type: Package
Version: 0.2-2
Date: 2012-01-22
Depends: amap, diagram
Suggests: FactoMineR
License: GPL (>= 2)
LazyLoad: yes

The main functions are `plspm`, `plspm.groups`, `rebus.pls`, `rebus.test`, `plsreg1`, `plsreg2`, `plsca`, and `nipals`.

Author(s)

Author: Gaston Sanchez, Laura Trinchera
Maintainer: Gaston Sanchez <plspm@gmail.com>

References

<http://www.plsmodeling.com>

Tenenhaus, M. (1998) *La Regression PLS. Theorie et Pratique*. Editions TECHNIP, Paris.

Handbook of Partial Least Squares (2010). Editors: V. Esposito Vinzo, W.W. Chin, J. Henseler, H. Wang. Springer, Paris.

arizona

Arizona vegetation dataset

Description

This dataset gives the measurements of 16 vegetation communities 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.

Usage

arizona

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.

<i>Num</i>	<i>Variable</i>	<i>Description</i>	<i>Concept</i>
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
```

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.

<i>Num</i>	<i>Variable</i>	<i>Description</i>	<i>Concept</i>
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	CSH: 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: League Country	country of the team's league	none
12	Rank: Rank Position	final ranking position within its league	none

Source

League Day. <http://www.leagueday.com>

Statto. <http://www.statto.com>

References

Applying plsmp Beginners Guide. <http://www.scribd.com/plspm>

Examples

```
data(futbol)
futbol
```

 it.reb

Response Based Unit Segmentation (REBUS) Algorithm - Iterative Steps

Description

REBUS-PLS is an iterative algorithm for performing response based clustering in a PLS-PM framework. This function allows to perform the iterative steps of the REBUS-PLS Algorithm (Steps 5-9). 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

<code>pls</code>	Object of class "plspm" returned by plspm .
<code>hclus.res</code>	Object of class "hclust" returned by res.clus
<code>nk</code>	Number of classes to take into account. This value should be defined according to the dendrogram obtained by performing res.clus .
<code>Y</code>	Optional dataset (matrix or data frame) used when argument <code>dataset=NULL</code> inside <code>pls</code> .
<code>stop.crit</code>	Number indicating the stop criterion for the iterative algorithm. The author suggests using the threshold of less than 0.05% of units changing class from one iteration to the other as stopping rule.
<code>iter.max</code>	An integer indicating the maximum number of iterations. Default value = 100.

Details

This function allows us to perform Steps 5-9 and the iteration procedure of the REBUS-PLS Algorithm. Moreover, this function provides the final class membership for each unit and summary results for the final local models.

A threshold of 6 units per class is fixed. If there is a class with less than 6 units, the algorithm stops. For more details refer to [rebus.pls](#).

For expert users: if you want to test REBUS-PLS on several number of classes you need to run this function several times by changing `nk`.

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 "rebus", basically a list with the following elements:

<code>loadings</code>	Matrix of standardized loadings (i.e. correlations with LVs.) for each local model.
<code>path.coefs</code>	Matrix of path coefficients for each local model.
<code>quality</code>	Matrix containing the average communalities, the average redundancies, the R2 values, and the GoF index for each local model.
<code>segments</code>	Vector defining the class membership of each unit.
<code>origdata.clas</code>	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.

http://www.fedoa.unina.it/view/people/Trinchera,_Laura.html

See Also

[plspm](#), [rebus.pls](#), [res.clus](#)

Examples

```
## Not run:
## example of rebus analysis with simulated data
data(sim.data)
## First compute GLOBAL model
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(sim.data, inner=sim.inner,
                   outer=sim.outer, modes=sim.mod)

sim.global
## Then compute cluster analysis on residuals of global model
sim.res.clus <- res.clus(sim.global)
## To conclude run iteration algorithm
rebus.sim <- it.reb(sim.global, sim.res.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)

## End(Not run)
```

local.models

Calculates 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

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

Details

The function `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", the function `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. The list contains the following elements:

<code>glob.model</code>	PLS-PM of the global model
<code>loc.model.1</code>	PLS-PM of segment (class) 1
<code>loc.model.2</code>	PLS-PM of segment (class) 2
<code>loc.model.k</code>	PLS-PM of segment (class) k

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. See `examples` below.

Author(s)

Laura Trinchera, Gaston Sanchez

See Also

[rebus.pls](#)

Examples

```
## Not run:
## example of rebus analysis
data(sim.data)
## First compute GLOBAL model
sim.inner <- matrix(c(0,0,0,0,0,0,0,1,1,0),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(sim.data, inner=sim.inner,
                   outer=sim.outer, modes=sim.mod)

sim.global
## Then compute cluster on residual from global model
sim.res.clus <- res.clus(sim.global)
## To conclude run iteration algorithm
rebus.sim <- it.reb(sim.global, sim.res.clus, nk=2,
                  stop.crit = 0.005, iter.max = 100 )
## Computation of local models
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 customer'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
va11 First MV of the block Value
va12 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)
```

`nipals`*NIPALS: Non-linear Iterative Partial Least Squares*

Description

Principal Component Analysis with NIPALS algorithm

Usage

```
nipals(x, nc = 2, scaled = TRUE)
```

Arguments

<code>x</code>	A numeric matrix or data frame.
<code>nc</code>	Number of components kept in the results (by default 2)
<code>scaled</code>	A logical value indicating whether scaling data is performed (TRUE by default).

Details

The function `nipals` performs Principal Component Analysis of a data matrix that may contain missing data.

Value

An object of class "nipals", basically a list with the following elements:

<code>values</code>	The pseudo eigenvalues.
<code>scores</code>	The extracted scores.
<code>loadings</code>	The loadings.
<code>cor.sco</code>	Correlations between the variables and the scores.
<code>disto</code>	Squared distance of the observations to the origin.
<code>contrib</code>	Contributions of the observations (rows).
<code>cos</code>	Squared cosinus.
<code>dmod</code>	Distance to the Model.

When the analyzed data contain missing values, the help interpretation tools (e.g. `cor.sco`, `disto`, `contrib`, `cos`, `dmod`) may not be meaningful, that is to say, some of the results may not be coherent.

Author(s)

Gaston Sanchez

References

- Tenenhaus, M. (1998) *La Regression PLS. Theorie et Pratique*. Editions TECHNIP, Paris.
 Tenenhaus, M. (2007) *Statistique. Methodes pour decrire, expliquer et prevoir*. Dunod, Paris.

See Also

[plot.nipals](#)

Examples

```
## Not run:
## example of NIPALS algorithm
data(wines)
nip1 <- nipals(wines[,-1], nc=5)
plot(nip1)

## USArrests data vary
nip2 <- nipals(USArrests)
plot(nip2)

## End(Not run)
```

orange

Orange Juice dataset

Description

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

Usage

orange

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.

<i>Num</i>	<i>Variable</i>	<i>Description</i>	<i>Concept</i>
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 typicality	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 judgements 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 Estadística*, **27**, pp. 1-26.

Examples

```
data(orange)
orange
```

```
plot.nipals
```

Plot NIPALS basic results

Description

Plot method for objects of class "nipals"

Usage

```
## S3 method for class 'nipals'
plot(x, ...)
```

Arguments

x An object of class "nipals" returned by [nipals](#).
 ... Further arguments are ignored.

Details

The function `plot.nipals` displays three plots:

- 1) screeplots of the pseudo eigenvalues
- 2) the circle of correlations between variables and the first two components (scores)
- 3) the plot of the two first scores, and the plot of the two first loadings

Author(s)

Gaston Sanchez

See Also

[nipals](#)

Examples

```
## Not run:
## example of NIPALS algorithm
data(wines)
nip1 <- nipals(wines[,-1], nc=5)
plot(nip1)

## End(Not run)
```

plot.plsca

Plot PLSCA basic results

Description

Plot method for objects of class "plsca"

Usage

```
## S3 method for class 'plsca'
plot(x, ...)
```

Arguments

`x` An object of class "plsca" returned by [plsca](#).
`...` Further arguments are ignored.

Details

The function `plot.plsca` displays four plots:

- 1) the circles of correlations of the X-scores (T-components) and the Y-scores (U-components)
- 2) the plot of T-components (t1,t2) and the plot of U-components (u1,u1)
- 3) barplots of the explained variances and communalities
- 4) the dot charts with (t1,u1) and (t2,u2)

Author(s)

Gaston Sanchez

See Also[plsca](#)**Examples**

```
## Not run:
## example of PLSCA with the vehicles dataset
data(vehicles)
can <- plsca(vehicles[,1:12], vehicles[,13:16])
plot(can)

## End(Not run)
```

plot.plspm

*Plot plspm path diagrams***Description**

Plot method for objects of class "plspm"

Usage

```
## S3 method for class 'plspm'
plot(x, what = "inner", how = "joint", arr.pos = 0.5,
     box.prop = 0.5, box.cex = 1, cex.txt = 1, ...)
```

Arguments

x	An object of class "plspm" returned by plspm .
what	A character string indicating what models should be plotted. Options: "inner" (default value), "weights", "loadings", and "all".
how	A character string indicating how the diagrams should be plotted. Options: "joint" (default value), and "split".
arr.pos	Relative position of arrowheads on arrows.
box.prop	Length/width ratio of boxes.
box.cex	Relative size of text in boxes.
cex.txt	Relative size of text on arrows.
...	Further arguments are ignored.

Details

The function `plot.plspm` allows to display path diagrams of a `plspm` model.
 If `what="inner"` only a diagram of the inner model is displayed.
 If `what="weights"` diagrams of the outer weights for each block of variables are displayed.
 If `what="loadings"` diagrams of the loadings for each block of variables are displayed.
 If `what="all"` diagrams of the inner model, outer weights, and loadings are displayed.
 When `how="split"` all diagrams are displayed separately by each block of variables:

Note

Function `plot.plspm` is based on the function `plotmat` of package `diagram`.
<http://cran.r-project.org/web/packages/diagram/vignettes/diagram.pdf>

See Also

[plspm](#)

Examples

```
## Not run:
## typical example of PLS-PM in customer satisfaction analysis
## model with six LVs and reflective indicators
data(satisfaction)
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)
sat.outer <- list(1:5,6:10,11:15,16:19,20:23,24:27)
sat.mod <- rep("A",6) ## reflective indicators
res2 <- plspm(satisfaction, sat.inner, sat.outer, sat.mod, scheme="centroid",
             scaled=FALSE)
## plot diagram of the inner model
plot(res2)
## plot diagrams of both the inner model and outer model (loadings and weights)
plot(res2, what="all")

## End(Not run)
```

`plot.plspm.groups`

Plot results from a group comparison test in PLS-PM

Description

Plot method for objects of class `"plspm.groups"`

Usage

```
## S3 method for class 'plspm.groups'  
plot(x, ...)
```

Arguments

x An object of class "plspm.groups" returned by [plspm.groups](#).
... Further arguments are ignored.

Details

The function `plot.plspm.groups` displays a barplot with the path coefficients of the compared groups.

See Also

[plspm.groups](#), [plspm](#)

Examples

```
## Not run:  
## example with customer satisfaction analysis  
## group comparison based on the segmentation variable "gender"  
data(satisfaction)  
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)  
sat.outer <- list(1:5,6:10,11:15,16:19,20:23,24:27)  
sat.mod <- rep("A",6) ## reflective indicators  
pls <- plspm(satisfaction, sat.inner, sat.outer, sat.mod, scheme="factor",  
            scaled=FALSE)  
## permutation test with 100 permutations  
res.group <- plspm.groups(pls, satisfaction$gender,  
                          method="permutation", reps=100)  
  
res.group  
plot(res.group)  
  
## End(Not run)
```

`plot.plsreg1`*Plot PLSR1 basic results*

Description

Plot method for objects of class "plsreg1"

Usage

```
## S3 method for class 'plsreg1'  
plot(x, ...)
```

Arguments

`x` An object of class "plsreg1" returned by `plsreg1`.
`...` Further arguments are ignored.

Details

The function `plot.plsreg1` displays four plots:

- 1) the circle of correlations between the variables and the first two components (scores)
- 2) the plot of components (t1,t2) and the plot of components (t1,u1)
- 3) the plot of T2 Hotelling Confidence ellipse
- 4) the dot chart comparing the response variable (Y) with the predicted values (Y-predicted)

Author(s)

Gaston Sanchez

See Also

[plsreg1](#), [plsreg2](#)

Examples

```
## Not run:  
## example of plsreg1  
data(vehicles)  
pls1 <- plsreg1(vehicles[,1:12], vehicles[,13])  
plot(pls1)  
  
## End(Not run)
```

`plot.plsreg2`*Plot PLSR2 basic results*

Description

Plot method for objects of class "plsreg2"

Usage

```
## S3 method for class 'plsreg2'  
plot(x, ...)
```

Arguments

`x` An object of class "plsreg2" returned by `plsreg2`.
`...` Further arguments are ignored.

Details

The function `plot.plsreg2` displays four plots:

- 1) the circle of correlations between the variables and the first two components (scores)
- 2) the plot of weights and loadings (w,c)
- 3) the plot of components (t1,t2) and the plot of components (u1,u2)
- 4) the plot of components (t1,u1) and the plot of components (t2,u2)

Author(s)

Gaston Sanchez

See Also

[plsreg2](#), [plsreg1](#)

Examples

```
## Not run:  
## example of plsreg2  
data(vehicles)  
pls2 <- plsreg2(vehicles[,1:12], vehicles[,13:16])  
plot(pls2)  
  
## End(Not run)
```

 plsca

PLS-CA: Partial Least Squares Canonical Analysis

Description

Performs partial least squares canonical analysis for two blocks of data. Compared to PLSR2, the blocks of variables in PLS-CA play a symmetric role (i.e. there is neither predictors nor predictands)

Usage

```
plsca(X, Y, nc = NULL, scaled = TRUE)
```

Arguments

X	A numeric matrix or data frame (X-block).
Y	A numeric matrix or data frame (Y-block).
nc	The number of extracted PLS components (NULL by default)
scaled	A logical value indicating whether scaling data should be performed (TRUE by default).

Details

Arguments X and Y must contain more than one variable.

No missing data are allowed.

When nc=NULL the number of components is determined by taking the minimum between the number of columns from X and Y.

When scaled=TRUE the data is scaled to standardized values (mean=0, variance=1). Otherwise the data will only be centered (mean=0).

Value

An object of class "plsca", basically a list with the following elements:

x.scores	scores of the X-block (also referred to as T components).
x.wgs	weights of the X-block.
x.loads	loadings of the X-block.
y.scores	scores of the Y-block (also referred to as U components).
y.wgs	weights of the Y-block.
y.loads	loadings of the Y-block.
cor.xt	correlations between X and T.
cor.yu	correlations between Y and U.
cor.tu	correlations between T and U.
cor.xu	correlations between X and U.

cor.yt	correlations between Y and T.
R2X	explained variance of X by T.
R2Y	explained variance of Y by U.
com.xu	communality of X with U.
com.yt	communality of Y with T.

Author(s)

Gaston Sanchez

References

Tenenhaus, M. (1998) *La Regression PLS. Theorie et Pratique*. Editions TECHNIP, Paris.

See Also

[print.plsca,plot.plsca](#)

Examples

```
## Not run:  
## example of PLSCA with the vehicles dataset  
data(vehicles)  
can <- plsca(vehicles[,1:12], vehicles[,13:16])  
can  
plot(can)  
  
## End(Not run)
```

plspm

PLS-PM: Partial Least Squares Path Modeling

Description

Estimate path models with latent variables by partial least squares approach

Usage

```
plspm(x, inner, outer, modes = NULL, scheme = "centroid", scaled = TRUE, boot.val = FALSE,  
      br = NULL, plsr = FALSE, tol = 0.00001, iter = 100, dataset = TRUE)
```

```
plspm.fit(x, inner, outer, modes = NULL, scheme = "centroid", scaled = TRUE, tol = 0.00001, iter = 100)
```

Arguments

<code>x</code>	A numeric matrix or data frame containing the manifest variables.
<code>inner</code>	A square (lower triangular) boolean matrix representing the inner model (i.e. the path relationships between latent variables).
<code>outer</code>	List of vectors with column indices from <code>x</code> representing the outer model (i.e. which manifest variables correspond to the latent variables).
<code>modes</code>	A character vector indicating the type of measurement for each latent variable. "A" for reflective measurement or "B" for formative measurement (NULL by default).
<code>scheme</code>	A string of characters indicating the type of inner weighting scheme. Possible values are "centroid", "factor", or "path".
<code>scaled</code>	A logical value indicating whether scaling data is performed (TRUE by default).
<code>boot.val</code>	A logical value indicating whether bootstrap validation is performed (FALSE by default).
<code>br</code>	An integer indicating the number bootstrap resamples. Used only when <code>boot.val=TRUE</code> .
<code>pls</code>	A logical value indicating whether pls regression is applied (FALSE by default).
<code>tol</code>	Decimal value indicating the tolerance criterion for the iterations (<code>tol=0.00001</code> by default).
<code>iter</code>	An integer indicating the maximum number of iterations (<code>iter=100</code> by default).
<code>dataset</code>	A logical value indicating whether the data matrix should be retrieved (TRUE by default).

Details

The function `plspm` estimates a path model by partial least squares approach while providing all results.

The function `plspm.fit` performs the basic PLS algorithm and provides simple results (e.g. outer weights, LVs scores, path coefficients, R2, and loadings).

The argument `inner.mat` is a matrix of zeros and ones that indicates the structural relationships between latent variables. This must be a lower triangular matrix. `inner.mat` will contain a 1 when column `j` affects row `i`, 0 otherwise.

The argument `sets` is a list of vectors of indices indicating the sets of manifest variables associated to the latent variables. The length of `sets` must be equal to the number of rows of `inner.mat`.

The argument `modes` is a character vector indicating the type of measurement for each latent variable. A value of "A" is used when a latent variable has reflective manifest variables, and a value of "B" is used when the latent variable has formative manifest variables. The length of `modes` must be equal to the number of rows of `inner.mat` (i.e. equal to the length of `sets`).

The argument `scaled` is TRUE by default. This means that data in `x` is scaled to standardized values (i.e. mean=0 and variance=1, calculating the variance dividing by `N` instead of `N-1`). Unless the data

has the same scale for all variables, it is strongly recommended to keep this argument unchanged.

When bootstrap validation is performed, the default number of re-samples is 100. However, `br` can be specified in a range from 100 to 1000.

The argument `plsr` gives the option to calculate the path coefficients by means of pls regression.

The argument `tol` can be specified in a range from 0 to 0.001.

The minimum value of `iter` is 100.

Value

An object of class "plspm". When the function `plspm.fit` is called, it returns a list with basic results:

<code>outer.mod</code>	Results of the outer (measurement) model. Includes: outer weights, standardized loadings, communalities, and redundancies.
<code>inner.mod</code>	Results of the inner (structural) model. Includes: path coefficients and R-squared for each endogenous latent variable.
<code>latents</code>	Matrix of standardized latent variables (variance=1 calculated divided by N) obtained from centered data (mean=0).
<code>scores</code>	Matrix of latent variables used to estimate the inner model. If <code>scaled=FALSE</code> then scores are latent variables calculated with the original data (non-standardized). If <code>scaled=TRUE</code> then scores and latents have the same values.
<code>out.weights</code>	Vector of outer weights.
<code>loadings</code>	Vector of standardized loadings (i.e. correlations with LVs.)
<code>path.coefs</code>	Matrix of path coefficients (this matrix has a similar form as <code>inner.mat</code>).
<code>r.sqr</code>	Vector of R-squared coefficients.

If the function `plspm` is called, the previous list of results also contains the following elements:

<code>outer.cor</code>	Correlations between the latent variables and the manifest variables (also called crossloadings).
<code>inner.sum</code>	Summarized results by latent variable of the inner model. Includes: type of LV, type of measurement, number of indicators, R-squared, average communality, average redundancy, and average variance extracted
<code>effects</code>	Path effects of the structural relationships. Includes: direct, indirect, and total effects.
<code>unidim</code>	Results for checking the unidimensionality of blocks (These results are only meaningful for reflective blocks).
<code>gof</code>	Table with indexes of Goodness-of-Fit. Includes: absolute GoF, relative GoF, outer model GoF, and inner model GoF.

data	Data matrix containing the manifest variables used in the model. Only when dataset=TRUE.
boot	List of bootstrapping results; only available when argument boot.val=TRUE.

Author(s)

Gaston Sanchez

<http://www.scribd.com/plspm>**References**

- Tenenhaus, M., Esposito Vinzi, V., Chatelin Y.M., and Lauro, C. (2005) PLS path modeling. *Computational Statistics & Data Analysis*, **48**, pp. 159-205.
- Tenenhaus, M., and Pages, J. (2001) Multiple factor analysis combined with PLS path modelling. Application to the analysis of relationships between physicochemical variables, sensory profiles and hedonic judgements. *Chemometrics and Intelligent Laboratory Systems*, **58**, pp. 261-273.
- Tenenhaus, M., and Hanafi, M. A bridge between PLS path modeling and multi-block data analysis. *Handbook on Partial Least Squares (PLS): Concepts, methods, and applications*. Springer: In press.
- Lohmoller, J.-B. (1989) *Latent variables path modelin 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[print.plspm](#), [summary.plspm](#), [plot.plspm](#).**Examples**

```
## Not run:
## example of PLS-PM in ecological analysis
## model with three LVs and formative indicators
data(arizona)
ari.inner <- matrix(c(0,0,0,0,0,0,1,1,0),3,3,byrow=TRUE)
dimnames(ari.inner) <- list(c("ENV","SOIL","DIV"),c("ENV","SOIL","DIV"))
ari.outer <- list(c(1,2),c(3,4,5),c(6,7,8))
ari.mod <- c("B","B","B") ## formative indicators
res1 <- plspm(arizona, inner=ari.inner, outer=ari.outer, modes=ari.mod,
scheme="factor", scaled=TRUE, pls=TRUE)
res1
summary(res1)

## typical example of PLS-PM in customer satisfaction analysis
## model with six LVs and reflective indicators
```

```

data(satisfaction)
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)
sat.outer <- list(1:5,6:10,11:15,16:19,20:23,24:27)
sat.mod <- rep("A",6) ## reflective indicators
res2 <- plspm(satisfaction, sat.inner, sat.outer, sat.mod, scaled=FALSE, boot.val=TRUE)
summary(res2)
plot(res2)

## example of PLS-PM in sensory analysis
## estimate a path model for the orange juice data
data(orange)
senso.inner <- matrix(c(0,0,0,1,0,0,1,1,0),3,3,byrow=TRUE)
dimnames(senso.inner) <- list(c("PHYCHEM","SENSORY","HEDONIC"),
                             c("PHYCHEM","SENSORY","HEDONIC"))
senso.outer <- list(1:9,10:16,17:112)
senso.mod <- rep("A",3)
res3 <- plspm(orange, senso.inner, senso.outer, senso.mod,
             scheme="centroid", scaled=TRUE)
summary(res3)

## example of PLS-PM in multi-block data analysis
## estimate a path model for the wine data set
## requires package FactoMineR
library(FactoMineR)
data(wine)
SMELL <- c(0,0,0,0)
VIEW <- c(1,0,0,0)
SHAKE <- c(1,1,0,0)
TASTE <- c(1,1,1,0)
wine.inner <- rbind(SMELL,VIEW,SHAKE,TASTE)
wine.outer <- list(3:7,8:10,11:20,21:29)
wine.mods <- rep("A",4)
# using function plspm.fit (basic pls algorithm)
res4 <- plspm.fit(wine, wine.inner, wine.outer, wine.mods, scheme="centroid")
plot(res4, what="all", arr.pos=.4, box.prop=.4, cex.txt=.8)

## End(Not run)

```

plspm.groups

Group Comparison in PLS-PM

Description

This function performs a group comparison test for comparing path coefficients between two groups

Usage

```
plsmp.groups(pls, g, Y = NULL, method = "bootstrap", reps = NULL)
```

Arguments

pls	An object of class "plsmp" returned by the function <code>plsmp</code>
g	A 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	The method to be used in the test. Possible values are "bootstrap" or "permutation"
reps	An integer indicating the number of either bootstrap resamples or number of permutations. If NULL then reps=100

Details

The function `plsmp.groups` performs a two groups comparison test in PLS-PM for comparing path coefficients between two groups. For the moment, 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.

The null and alternative hypotheses to be tested are:

H0: path coefficients are not significantly different

H1: path coefficients are significantly different

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 "plsmp.groups", basically a list with the following elements:

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 results of the inner model for the global model
group1	List with results of the inner model for group1
group2	List with results of the inner model for group2

Author(s)

Gaston Sanchez

References

Chin, W.W. (2003) A permutation procedure for multi-group comparison of PLS models. In: Vi-lares, 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. Available from: <http://disc-nt.cba.uh.edu/chin/plsfaq/multigroup.htm>

See Also

[print.plspm.groups](#), [plot.plspm.groups](#)

Examples

```
## Not run:
## example with customer satisfaction analysis
## group comparison based on the segmentation variable "gender"
data(satisfaction)
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)
sat.outer <- list(1:5,6:10,11:15,16:19,20:23,24:27)
sat.mod <- rep("A",6) ## reflective indicators
pls <- plspm(satisfaction, sat.inner, sat.outer, sat.mod, scheme="factor",
            scaled=FALSE)
## permutation test with 100 permutations
res.group <- plspm.groups(pls, satisfaction$gender,
                        method="permutation", reps=100)
res.group
plot(res.group)

## End(Not run)
```

plsreg1

PLS-R1: Partial Least Squares Regression 1

Description

Calculates partial least squares regression for the univariate case (i.e. one response variable)

Usage

```
plsreg1(x, y, nc = 2, cv = FALSE)
```

Arguments

x	A numeric matrix or data frame with the predictor variables (which may contain missing data).
y	A numeric vector for the response or predictand variable.
nc	The number of extracted PLS components (2 by default).
cv	A logical value indicating whether cross-validation should be performed (FALSE by default).

Details

The minimum number of PLS components `nc` to be extracted is 2.

The argument `x` may contain missing data. Conversely, the argument `y` must not contain missing values.

The data is scaled to standardized values (mean=0, variance=1).

The argument `cv` gives the option to perform leave-one-out cross validation to choose the best number of PLS components.

Value

An object of class "plsreg1", basically a list with the following elements:

scores	PLS components.
x.loads	loadings of the predictor variables.
y.loads	loadings of the predictand variable.
u.scores	u scores of the predictand variable.
raw.wgs	weights to calculate the PLS scores with the deflated matrices of predictor variables.
mod.wgs	modified weights to calculate the PLS scores with the matrix of predictor variables.
std.coef	Vector of standardized regression coefficients.
coeffs	Vector of regression coefficients (used with the original data scale).
R2	Vector of PLS R-squared.
y.pred	Vector of predicted values.
resid	Vector of residuals.
cor.sco	Correlations between the variables and the PLS components.
T2	Table of Hotelling T2 values (used to detect atypical observations).
Q2	Table with the cross validation results. Includes: PRESS, RSS, Q2, and cumulated Q2. Only available when <code>cv=TRUE</code>

Author(s)

Gaston Sanchez

References

- Geladi, P., and Kowalski, B. (1986) Partial Least Squares Regression: A Tutorial. *Analytica Chimica Acta*, **185**, pp. 1-17.
- Tenenhaus, M. (1998) *La Regression PLS. Theorie et Pratique*. Editions TECHNIP, Paris.
- Tenenhaus, M., Gauchi, J.-P., and Menardo, C. (1995) Regression PLS et applications. *Revue de statistique appliquee*, **43**, pp. 7-63.
- Valencia, J.L., Diaz-Llanos, F.J. (2004) *Metodos de Prediccion en Situaciones Limite*. Editorial La Muralla, S.A. Madrid.

See Also

[print.plsreg1](#), [plot.plsreg1](#), [plsreg2](#).

Examples

```
## Not run:  
## example of PLSR1 with the vehicles dataset  
## predictand variable: price of vehicles  
data(vehicles)  
pls1 <- plsreg1(vehicles[,1:12], vehicles[,13], cv=TRUE)  
pls1  
plot(pls1)  
  
## End(Not run)
```

plsreg2

PLS-R2: Partial Least Squares Regression 2

Description

Calculates partial least squares regression for the multivariate case (i.e. more than one response variable)

Usage

```
plsreg2(X, Y, nc = 2)
```

Arguments

- | | |
|----|---|
| X | A numeric matrix or data frame containing the predictor variables. |
| Y | A numeric matrix or data frame containing the predictand variables. |
| nc | The number of extracted PLS components (2 by default) |

Details

The minimum number of PLS components `nc` is 2.

The data is scaled to standardized values (mean=0, variance=1).

No missing data are allowed.

Argument `Y` must contain more than one variable. If `Y` is a vector, you may use the function `plsreg1`.

Value

An object of class "plsreg2", basically a list with the following elements:

<code>x.scores</code>	components of the predictor variables.
<code>x.loads</code>	loadings of the predictor variables.
<code>y.scores</code>	components of the predictand variables.
<code>y.loads</code>	loadings of the predictand variables.
<code>raw.wgs</code>	weights to calculate the PLS scores with the deflated matrices of predictor variables.
<code>mod.wgs</code>	modified weights to calculate the PLS scores with the matrix of predictor variables.
<code>cor.tx</code>	modified weights to calculate the PLS scores with the matrix of predictor variables.
<code>cor.ty</code>	modified weights to calculate the PLS scores with the matrix of predictor variables.
<code>std.coef</code>	Vector of standardized regression coefficients.
<code>coeffs</code>	Vector of regression coefficients (used with the original data scale).
<code>y.pred</code>	Vector of predicted values.
<code>resid</code>	Vector of residuals.
<code>expvar</code>	table with R-squared coefficients.
<code>Q2</code>	table of Q2 indexes (i.e. leave-one-out cross validation).
<code>Q2cum</code>	table of cumulated Q2 indexes.
<code>VIP</code>	Variable Importance for Projection.

Author(s)

Gaston Sanchez

References

- Geladi, P., and Kowalski, B. (1986) Partial Least Squares Regression: A Tutorial. *Analytica Chimica Acta*, **185**, pp. 1-17.
- Hoskuldsson, A. (1988) PLS Regression Methods. *Journal of Chemometrics*, **2**, pp. 211-228.
- Tenenhaus, M. (1998) *La Regression PLS. Theorie et Pratique*. Editions TECHNIP, Paris.
- Valencia, J.L., Diaz-Llanos, F.J. (2004) *Metodos de Prediccion en Situaciones Limite*. Editorial La Muralla, S.A. Madrid.

See Also

[print.plsreg2](#), [plot.plsreg2](#), [plsreg1](#).

Examples

```
## Not run:  
## example of PLSR2 with the vehicles dataset  
data(vehicles)  
pls2 <- plsreg2(vehicles[,1:12], vehicles[,13:16])  
pls2  
plot(pls2)  
  
## End(Not run)
```

`print.local.models` *Printing local.models objects*

Description

Printing method for objects of class "local.models"

Usage

```
## S3 method for class 'local.models'  
print(x, ...)
```

Arguments

`x` Object of class "local.models" returned by [local.models](#).
`...` Further arguments are ignored.

Details

The function `local.models` displays the elements of a list containing all the results.

See Also

[local.models](#)

print.nipals	<i>Printing nipals obejcts</i>
--------------	--------------------------------

Description

Printing method for objects of class "nipals"

Usage

```
## S3 method for class 'nipals'  
print(x, ...)
```

Arguments

x	Object of class "nipals" returned by nipals .
...	Further arguments are ignored.

Details

The function `print.nipals` displays the elements of a list containing all the results.

See Also

[nipals,plot.nipals](#)

print.plsca	<i>Printing plsca objects</i>
-------------	-------------------------------

Description

Printing method for objects of class "plsca"

Usage

```
## S3 method for class 'plsca'  
print(x, ...)
```

Arguments

x	Object of class "plsca" returned by plsca .
...	Further arguments are ignored.

Details

The function `print.plsca` displays the elements of a list containing all the results.

See Also[plsca.plot.plsca](#)

print.plspm	<i>Printing plspm objects</i>
-------------	-------------------------------

Description

Print method for objects of class "plspm"

Usage

```
## S3 method for class 'plspm'  
print(x, ...)
```

Arguments

x	Object of class "plspm" returned by plspm .
...	Further arguments are ignored.

Details

The function `print.plspm` displays the elements of a list containing all the results.

See Also[plspm](#)

print.plspm.groups	<i>Printing plspm.groups objects</i>
--------------------	--------------------------------------

Description

Printing method for objects of class "plspm.groups"

Usage

```
## S3 method for class 'plspm.groups'  
print(x, ...)
```

Arguments

x	Object of class "plspm.groups" returned by plspm.groups .
...	Further arguments are ignored.

Details

The function `print.plspm.groups` displays a list with the results of the test.

See Also

[plspm.groups](#), [plspm](#)

`print.plsreg1` *Printing plsreg1 objects*

Description

Printing method for objects of class "plsreg1"

Usage

```
## S3 method for class 'plsreg1'
print(x, ...)
```

Arguments

`x` Object of class "plsreg1" returned by [plsreg1](#).
`...` Further arguments are ignored.

Details

The function `print.plsreg1` displays the elements of a list containing all the results.

See Also

[plsreg1](#), [plot.plsreg1](#)

`print.plsreg2` *Printing plsreg2 objects*

Description

Printing method for objects of class "plsreg2"

Usage

```
## S3 method for class 'plsreg2'
print(x, ...)
```

Arguments

- x Object of class "plsreg2" returned by [plsreg2](#).
- ... Further arguments are ignored.

Details

The function `print.plsreg2` displays the elements of a list containing all the results.

See Also

[plsreg2](#), [plot.plsreg2](#)

`print.rebus` *Printing rebus objects*

Description

Printing method for objects of class "rebus"

Usage

```
## S3 method for class 'rebus'  
print(x, ...)
```

Arguments

- x Object of class "rebus" returned by either [it.reb](#) or [rebus.pls](#).
- ... Further arguments are ignored.

Details

The function `print.rebus` displays the elements of a list containing all the results.

See Also

[rebus.pls](#)

print.rebus.test *Printing rebus.test objects*

Description

Printing method for objects of class "rebus.test"

Usage

```
## S3 method for class 'rebus.test'  
print(x, ...)
```

Arguments

x Object of class "rebus.test" returned by [rebus.test](#).
... Further arguments are ignored.

Details

The function `print.rebus.test` displays the names of the elements from a list containing all the results for each pair of groups comparison.

See Also

[rebus.test](#), [rebus.pls](#)

print.summary.plspm *Printing summaries of plspm objects*

Description

Print method of summary for objects of class "plspm"

Usage

```
## S3 method for class 'summary.plspm'  
print(x, ...)
```

Arguments

x Object of class "summary.plspm".
... Further arguments are ignored.

See Also

[plspm](#), [summary.plspm](#).

rebus.pls

*Response Based Unit Segmentation (REBUS) Algorithm - complete -***Description**

REBUS-PLS is an iterative algorithm for performing response based clustering in a PLS-PM framework.

This function allows to perform all the steps of the REBUS-PLS Algorithm.

Usage

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

Arguments

<code>pls</code>	Object of class "plspm" returned by <code>plspm</code> .
<code>Y</code>	Optional dataset (matrix or data frame) used when argument <code>dataset=NULL</code> inside <code>pls</code> .
<code>stop.crit</code>	Number indicating the stop criterion for the iterative algorithm. The author suggests using the threshold of less than 0.05% of units changing class from one iteration to the other as stopping rule.
<code>iter.max</code>	An integer indicating the maximum number of iterations. Default value = 100.

Details

REBUS-PLS is an iterative algorithm that starting from the global model allows us to detect local models performing better than the global model. The algorithm is composed by 9 steps:

Step 1: The first step of the REBUS-PLS algorithm involves computing the global model on all the observed units, by performing a simple PLS Path Modeling analysis (`plspm`).

Steps 2 and 3: In the second step, the communality and the structural residuals of each unit from the global model are obtained. The number of classes (K) to be taken into account during the successive iterations and the initial composition of the classes are obtained by performing a Hierarchical Cluster Analysis on the computed residuals (both from the measurement and the structural models).

Step 4: Choose the number of classes (K) to take into account according to the dendrogram obtained at Step 3.

Step 5: Once the number of classes to consider is chosen, the initial composition of the classes are obtained according to results of the Hierarchical Cluster Analysis.

Step 6: A PLS Path Modeling analysis is performed on each formed class and K provisional local models are estimated.

The group-specific parameters computed at the previous step are used to compute the communality and the structural residuals of each unit from each local model.

Step 7: The Closeness Measure (CM) of each unit from each local model is computed.

Step 8: Each unit is, therefore, assigned to the closest local model, i.e. to the model from which each unit shows the smallest CM value.

Step 9: Once the composition of the classes is updated, K new local models are estimated.

Step 5-9 have to be iterate until there is convergence on class composition. The author suggests using the threshold of less than 0.05% of units changing class from one iteration to the other as stopping rule.

In addition, a threshold of 6 units per class is fixed. If there is a class with less than 6 units, the algorithm stops.

This function allows us to perform in a unique function all the steps of the REBUS-PLS Algorithm. In particular, firstly it computes the communality and structural residuals from the global model and performs a Hierarchical Cluster Analysis on this residuals (by applying `res.clus` function). Then it shows the obtained dendrogram and asks the user to choose the number of clusters to be take into account. This value need to be an integer value higher than 1. Once the number of classes is chosen, this function performs the iteration steps of the REBUS-PLS Algorithm and directly provides the final class membership for each unit, summary results for the final local models, and the Group Quality Index (GQI) (Trinchera, 2007) ((by applying `it.reb` function).

Once the convergence is achieved you should:

- (i) Estimate the final local models
- (ii) Validate the Group Quality Index

Both these last two steps can be performed by running `local.models`.

N.B. Once the global model is computed it is possible to obtain the same results as the ones obtained by running `rebus.pls` function, by running the function `res.clus` and then the function `it.reb`. If you want to test REBUS-PLS on several number of classes you need to run `it.reb` function several times by changing `nk`. This allows you to avoid computing the residuals each time from the global model.

Value

An object of class "rebus", basically a list with the following elements:

<code>loadings</code>	Matrix of standardized loadings (i.e. correlations with LVs.) for each local model.
<code>path.coefs</code>	Matrix of path coefficients for each local model.
<code>quality</code>	Matrix containing the average communalities, the average redundancies, the R2 values, and the GoF values for each local model.
<code>segments</code>	Vector defining for each unit the class membership.
<code>origdata.clas</code>	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.

http://www.fedoa.unina.it/view/people/Trinchera,_Laura.html

See Also

[plspm](#), [res.clus](#), [resclus.plot](#), [it.reb](#), [rebus.test](#), [local.models](#)

Examples

```
## Not run:
## example of rebus analysis with simulated data
data(sim.data)
## First compute GLOBAL model
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(sim.data, inner=sim.inner,
                   outer=sim.outer, modes=sim.mod)

sim.global
## run rebus.pls function on the sim.data 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)
##
## ONCE COMPUTED THE GLOBAL MODEL IT IS POSSIBLE TO OBTAIN THE SAME RESULTS AS THE ONES
## OBTAINED BY RUNNING rebus.pls FUNCTION BY RUNNING THE FUNCTION res.clus
## AND THEN THE FUNCTION it.reb.
##
## Example:
## Once the global model is calculated you need to perform cluster analysis on residuals of the global model:
sim.res.clus <- res.clus(sim.global)
## and then run the iteration algorithm:
rebus.sim.twofunc <- it.reb(sim.global, sim.res.clus, nk=2,
                          stop.crit=0.005, iter.max=100)
##
## rebus.sim = rebus.sim.twofunc

## End(Not run)
```

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. The number of classes handled by `rebus.test` is limited to 6.

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 "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: Vi-lares, 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:
## example of rebus analysis with simulated data
data(sim.data)
## Calcualte 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(sim.data, inner=sim.inner,
                   outer=sim.outer, modes=sim.mod)

sim.global
## Cluster analysis on residuals of global model
sim.res.clus <- res.clus(sim.global)
## Iterative steps of REBUS algorithm on 2 classes
rebus.sim <- it.reb(sim.global, sim.res.clus, nk=2,
                  stop.crit=0.005, iter.max=100)
sim.permu <- rebus.test(sim.global, rebus.sim)
sim.permu
sim.permu$test_1_2
## or equivalently
sim.permu[[1]]

## End(Not run)
```

res.clus	<i>Cluster Analysis on communality and structural residuals of a PLS-PM object</i>
----------	--

Description

This function computes communality and structural residuals from the global 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" returned by <code>plspm</code> .
Y	Optional dataset (matrix or data frame) used when argument <code>dataset=NULL</code> inside <code>pls</code> .

Details

The function `res.clus` comprises the second and third steps of the REBUS-PLS Algorithm. In particular it computes communality and structural residuals (as defined in Trinchera (2007) and Esposito Vinzi *et al.* (2008)) of each unit from the global model (step two of REBUS-PLS Algorithm). 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.

The number of classes (K) to be taken into account during the successive steps of the REBUS-PLS Algorithm (performed by `it.reb`), and the initial composition of the classes are obtained according to the results of the Hierarchical Cluster Analysis. Users must choose K according to this dendrogram and use it as an argument in the `it.reb` function.

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 "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

`resclus.plot`, `it.reb`, `plspm`

Examples

```
## Not run:
## example of rebus analysis with simulated data
data(sim.data)
## First compute GLOBAL model
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(sim.data, inner=sim.inner,
                   outer=sim.outer, modes=sim.mod)
```

```

sim.global
## Then compute cluster analysis on the residuals of global model
sim.res.clus <- res.clus(sim.global)

## End(Not run)

```

resclus.plot	<i>Colored dendrogram for communality and structural residuals according to the REBUS algorithm</i>
--------------	---

Description

Plot function for objects of class "hclust" returned by res.clus

Usage

```

resclus.plot(x, k=2, col.up="black", col.down=rainbow(k), lty.up=2, lty.down=1, lwd.up=1, lwd.down=1,
             type="rectangle", knot.pos="mean", show.labels=FALSE, only.tree=FALSE, members, ...)

```

Arguments

x	An object of class "hclust" obtained from res.clus.
k	The number of classes.
col.up	The color to be used for the lines above the cut of the dendrogram. Default "black".
col.down	The colors to be used for the classes below the cut of the dendrogram.
lty.up	The line type above the cut of the dendrogram. Line types are specified as an integer (0=blank, 1=solid (default), 2=dashed, 3=dotted, 4=dotdash, 5=longdash, 6=twodash).
lty.down	The line type for the classes below the cut of the dendrogram.
lwd.up	The line width above the cut of the dendrogram. Line width must be a positive number, defaulting to 1.
lwd.down	The line width for the classes (below the cut of the dendrogram).
type	The type of dendrogram. Can be either "rectangle" or "triangle".
knot.pos	Position of the knots. Can be "mean", "bary", "left", "right" or "random".
show.labels	Logical value indicating whether the labels of the objects should be printed. Default FALSE.
only.tree	Logical value indicating whether only the dendrogram should be printed. Default FALSE.
members	NULL or a vector with length size of a dissimilarity structure as produced by dist .
...	Further arguments are ignored.

Details

The function `resclus.plot` displays a dendrogram that helps to visualize the partitions in a dendrogram from `res.clus` results.

References

Modified version of the functions created by Romain Fracois. Source code available at: <http://addictedtor.free.fr/graphiques/RGraphGallery.php?graph=79>

See Also

[res.clus](#), [it.reb](#)

Examples

```
## Not run:
## example of rebus analysis with simulated data
data(sim.data)
## First compute GLOBAL model
sim.mat <- matrix(c(0,0,0,0,0,0,1,1,0),3,3,byrow=TRUE)
dimnames(sim.mat) <- list(c("Price","Quality","Satisfaction"),
                          c("Price","Quality","Satisfaction"))
sim.sets <- 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(sim.data, sim.mat, sim.sets, sim.mod)
sim.global
## Then compute cluster analysis on the residuals of global model
sim.res.clus <- res.clus(sim.global)
## Plot dendrogram with 4 classes specifying colors
dev.new()
resclus.plot(sim.res.clus, k=4, col.up="gray", col.down=c("orange","royalblue","green3","red"))
## Plot triangular dendrogram with 4 classes, labels shown, and wider lines
dev.new()
resclus.plot(sim.res.clus, k=4, lwd.down=2, col.up="gray", show.labels=TRUE, type="triangle")

## End(Not run)
```

ropes

Climbing Ropes dataset

Description

This dataset gives the measurements of 101 climbing ropes available in the market by spring 2011. The data was collected from the brands websites.

Usage

```
data(ropes)
```

Format

A data frame with 101 observations on the following 7 variables.

brand a factor with the brand names
 diameter the diameter of the ropes measured in mm
 weight the weight measured in grams per meter
 falls the number of UIAA falls
 imp.force the impact force value
 stat.elong the static elongation value
 dyn.elong the dynamic elongation value

References

Sanchez, G. (2012) PLS Modeling (In Press)

Examples

```
data(ropes)
summary(ropes)
```

 russett

Russett dataset

Description

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

Usage

```
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.

AGRIN: Recommended variables gini, farm, rent.

INDEV: Recommended variables gnpr, labo.

POLINS: Recommended variables inst, ecks, death, demostab, demoinst, dictator.

gini Inequality of land distribution
 farm Percentage of farmers that own half of the land
 rent Percentage of farmers that rent all their land

gnpr Gross national product per capita
 labo Percentage of labor force employed in agriculture
 inst Instability of executive (45-61)
 ecks Number of violent internal war incidents (41-61)
 death Number of people killed as a result of civic group violence (50-62)
 demostab Political regime: stable democracy
 demoinst Political regiem: unstable democracy
 dictator Political regime: dictatorship

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	<i>Satisfaction dataset</i>
--------------	-----------------------------

Description

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

Usage

```
satisfaction
```

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
```

sim.data

Simulated data for REBUS with two groups

Description

Simulated data with two latent classes showing different local models.

Usage

```
data(sim.data)
```

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) STEM: 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(sim.data)
sim.data
```

spainfoot

Spanish football dataset

Description

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

Usage

spainfoot

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.

ATTACK: Recommended variables GSH, GSA, SSH, SSA.

DEFENSE: Recommended variables GCH, GCA, CSH, CSA.

SUCCESS: Recommended variables WMH, WMA, LWR, LRWL.

INDIS: Recommended variables YC and RC.

GSH Goals Scored Home: total number of goals scored at home

GSA Goals Scored Away: total number of goals scored away

SSH Success to Score Home: Percentage of matches with scores goals at home

SSA Success to Score Away: Percentage of matches with scores goals away

GCH Goals Conceded Home: total number of goals conceded at home

GCA Goals Conceded Away: total number of goals conceded away

CSH Clean Sheets Home: percentage of matches with no conceded goals at home

CSA Clean Sheets Away: percentage of matches with no conceded goals away

WMH Won Matches Home: total number of won matches at home

WMA Won matches Away: total number of won matches away

LWR Longest Winning Run: longest run of won matches

LRWL Longest Run Without Loss: longest run of matches without losing

YC Yellow Cards: total number of yellow cards

RC Red Cards: total number of red cards

Source

League Day. <http://www.leagueday.com>

Statto. <http://www.statto.com>

BDFutbol. <http://www.statto.com>

Cero a cero. <http://www.ceroacero.es/>

References

Applying plspm Beginners Guide. <http://www.scribd.com/plspm>

Examples

```
data(spainfoot)
spainfoot
```

summary.plspm

Summarizing partial least squares path modeling analysis

Description

Summary method for objects of class "plspm".

Usage

```
## S3 method for class 'plspm'
summary(object, ...)
```

Arguments

object	Object of class "plspm".
...	Further arguments are ignored.

Details

The function `summary.plspm` gives the detailed numerical results of the `plspm` function.

First, the model specification is shown followed by the definition of the blocks of variables.

In second place the unidimensionality results of the blocks of manifest variables are displayed.

Then the outer (measurement) model and crossloadings are presented.

In fourth place, the inner (structural) model results as well as the correlations among latent variables are shown.

Next, a summary of the inner model and the goodness-of-fit indexes are presented.

Finally, a table with the path effects of the structural relationships is given.

In addition, if bootstrap validation was performed, the bootstrapping results are listed.

vehicles *Vehicles dataset*

Description

These data are the specification of 30 vehicles in terms of various characteristics.

Usage

vehicles

Format

A data frame with 30 observations and 16 variables.

<i>Num</i>	<i>Variable</i>	<i>Description</i>
1	diesel	Diesel fuel-type
2	turbo	Turbo aspiration
3	two.doors	Vehicles with two doors
4	hatchback	Hatchback body-style
5	wheel.base	Wheel base
6	length	Length
7	width	Width
8	height	Height
9	curb.weight	Curb weight
10	eng.size	Engine size
11	horsepower	Horsepower
12	peak.rpm	Peak revolutions per minute
13	price	Price in dollars
14	symbol	Insurance risk rating
15	city.mpg	Fuel consume in city
16	highway.mpg	Fuel consume in highway

Source

- 1) 1985 Model Import Car and Truck Specifications, 1985 Ward's Automotive Yearbook.
- 2) Personal Auto Manuals, Insurance Services Office, 160 Water Street, New York, NY 10038.
- 3) Insurance Collision Report, Insurance Institute for Highway Safety, Watergate 600, Washington, DC 20037.

Machine Learning Repository. <http://archive.ics.uci.edu/ml/datasets/Automobile>

Examples

```
data(vehicles)
vehicles
```

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.

Usage

wines

Format

A data frame with 178 observations and 14 variables.

<i>Num</i>	<i>Variable</i>	<i>Description</i>
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. <http://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

*Topic **datasets**

- arizona, [3](#)
- futbol, [4](#)
- mobile, [9](#)
- orange, [12](#)
- ropes, [44](#)
- russett, [45](#)
- satisfaction, [46](#)
- sim.data, [47](#)
- spainfoot, [49](#)
- vehicles, [51](#)
- wines, [52](#)

*Topic **package**

- plsmpm-package, [2](#)

arizona, [3](#)

dist, [43](#)

futbol, [4](#)

it.reb, [5](#), [35](#), [38–40](#), [42](#), [44](#)

local.models, [7](#), [31](#), [38–40](#)

mobile, [9](#)

nipals, [3](#), [11](#), [13](#), [14](#), [32](#)

orange, [12](#)

plot.nipals, [12](#), [13](#), [32](#)

plot.plsca, [14](#), [21](#), [33](#)

plot.plspm, [15](#), [24](#)

plot.plspm.groups, [16](#), [27](#)

plot.plsreg1, [18](#), [29](#), [34](#)

plot.plsreg2, [19](#), [31](#), [35](#)

plotmat, [16](#)

plsca, [3](#), [14](#), [15](#), [20](#), [32](#), [33](#)

plsmpm, [3](#), [6](#), [7](#), [15–17](#), [21](#), [26](#), [33](#), [34](#), [36](#), [37](#),
[39–42](#), [50](#)

plsmpm-package, [2](#)

plsmpm.groups, [3](#), [17](#), [25](#), [33](#), [34](#)

plsreg1, [3](#), [18](#), [19](#), [27](#), [30](#), [31](#), [34](#)

plsreg2, [3](#), [18](#), [19](#), [29](#), [29](#), [35](#)

print.local.models, [31](#)

print.nipals, [32](#)

print.plsca, [21](#), [32](#)

print.plspm, [24](#), [33](#)

print.plspm.groups, [27](#), [33](#)

print.plsreg1, [29](#), [34](#)

print.plsreg2, [31](#), [34](#)

print.rebus, [35](#)

print.rebus.test, [36](#)

print.summary.plspm, [36](#)

rebus.pls, [3](#), [6–8](#), [35](#), [36](#), [37](#), [40](#)

rebus.test, [3](#), [36](#), [39](#), [39](#)

res.clus, [5–7](#), [38](#), [39](#), [41](#), [44](#)

resclus.plot, [39](#), [42](#), [43](#)

ropes, [44](#)

russett, [45](#)

satisfaction, [46](#)

sim.data, [47](#)

spainfoot, [49](#)

summary.plspm, [24](#), [36](#), [50](#)

vehicles, [51](#)

wines, [52](#)