Package ‘ropls’

January 15, 2017

Type Package

Title PCA, PLS(-DA) and OPLS(-DA) for multivariate analysis and feature selection of omics data

Version 1.6.2

Date 2016-11-01

Author Etienne A. Thevenot <etienne.thevenot@cea.fr>

Maintainer Etienne A. Thevenot <etienne.thevenot@cea.fr>

biocViews Regression, Classification, PrincipalComponent, Transcriptomics, Proteomics, Metabolomics, Lipidomics, MassSpectrometry

Description Latent variable modeling with Principal Component Analysis (PCA) and Partial Least Squares (PLS) are powerful methods for visualization, regression, classification, and feature selection of omics data where the number of variables exceeds the number of samples and with multicollinearity among variables. Orthogonal Partial Least Squares (OPLS) enables to separately model the variation correlated (predictive) to the factor of interest and the uncorrelated (orthogonal) variation. While performing similarly to PLS, OPLS facilitates interpretation. Successful applications of these chemometrics techniques include spectroscopic data such as Raman spectroscopy, nuclear magnetic resonance (NMR), mass spectrometry (MS) in metabolomics and proteomics, but also transcriptomics data. In addition to scores, loadings and weights plots, the package provides metrics and graphics to determine the optimal number of components (e.g. with the R2 and Q2 coefficients), check the validity of the model by permutation testing, detect outliers, and perform feature selection (e.g. with Variable Importance in Projection or regression coefficients). The package can be accessed via a user interface on the Workflow4Metabolomics.org online resource for computational metabolomics (built upon the Galaxy environment).

Imports Biobase, methods

Suggests BiocGenerics, BiocStyle, CAMERA, faahKO, knitr, multtest, markdown, RUnit, xcms

VignetteBuilder knitr

License CeCILL

Lazy data yes

URL http://dx.doi.org/10.1021/acs.jproteome.5b00354

NeedsCompilation no

RoxygenNote 5.0.1
ropls-package

R topics documented:

ropls-package ................................................. 2
aminoacids ..................................................... 3
cellulose ....................................................... 4
checkW4M ....................................................... 5
cornell ......................................................... 6
fitted,opls-method ........................................... 7
foods ............................................................ 8
fromW4M ......................................................... 9
g>LoadingMN ..................................................... 9
g getPcaVarVn .................................................. 10
gScoreMN ....................................................... 11
gSubsetVi ....................................................... 12
gSummaryDF ................................................... 13
gVipVn .......................................................... 14
gWeightMN ...................................................... 15
linnerud ......................................................... 16
lowarp .......................................................... 16
mark ............................................................. 17
opls .............................................................. 18
opls-class ...................................................... 18
plot,opls-method .............................................. 24
predict,opls-method .......................................... 26
print,opls-method ............................................. 27
residuals,opls-method ....................................... 27
sacurine ......................................................... 28
show,opls-method ............................................. 29
strF .............................................................. 30
tested .......................................................... 30
toW4M .......................................................... 31

Index 33

ropls-package  PCA, PLS(-DA) and OPLS(-DA) for multivariate analysis and feature selection of omics data

Description

Latent variable modeling with Principal Component Analysis (PCA) and Partial Least Squares (PLS) are powerful methods for visualization, regression, classification, and feature selection of omics data where the number of variables exceeds the number of samples and with multicollinearity among variables. Orthogonal Partial Least Squares (OPLS) enables to separately model the variation correlated (predictive) to the factor of interest and the uncorrelated (orthogonal) variation. While performing similarly to PLS, OPLS facilitates interpretation. Successful applications of these chemometrics techniques include spectroscopic data such as Raman spectroscopy, nuclear magnetic resonance (NMR), mass spectrometry (MS) in metabolomics and proteomics, but also transcriptomics data. In addition to scores, loadings and weights plots, the package provides metrics and graphics to determine the optimal number of components (e.g. with the R2 and Q2 coefficients),
check the validity of the model by permutation testing, detect outliers, and perform feature selection (e.g. with Variable Importance in Projection or regression coefficients). The package can be accessed via a user interface on the Workflow4Metabolomics.org online resource for computational metabolomics (built upon the Galaxy environment).

**Author(s)**
E. Thevenot (CEA, LIST, MetaboHub)
Maintainer: Etienne Thevenot <etienne.thevenot@cea.fr>

**Examples**

```r
data(sacurine)
attach(sacurine)

#### PCA
sacurine.pca <- opls(dataMatrix)
plot(sacurine.pca, parAsColFcVn = sampleMetadata[, "gender"])

#### PLS-DA
sacurine.plsda <- opls(dataMatrix, sampleMetadata[, "gender"])

#### OPLS-DA
sacurine.oplsda <- opls(dataMatrix, sampleMetadata[, "gender"], predI = 1, orthoI = NA)
detach(sacurine)
```

---

**aminoacids**

_Amino-Acids Dataset_

**Description**
Quantitative structure property relationship (QSPR)

**Format**
A data frame with the following parameters:

- AA amino acid
- PIE lipophilicity constant of the AA side chain
- PIF lipophilicity constant of the AA side chain
- DGR free energy of transfer of an AA side chain from protein interior to water
- SAC water-accessible surface area of AA's calculated by MOLSV
- MR molecular refractivity
- Lam polarity parameter
- Vol molecular volume of AA's calculated by MOLSV
- DDGTS free energy of unfolding of the tryptophane synthase a unit of bacteriophage T4 lysosome
Value

Data frame (numeric type except the first column, which can be transformed into row names) with 19 rows and the 9 columns containing information about amino acids. For details see the 'Format' section above.

Source

'aminoacids' dataset.

References


cellulose

NIR-Viscosity example data set to illustrate multivariate calibration using PLS, spectral filtering and OPLS

Description

The data were collected at Akzo Nobel, Ornkoldsvik (Sweden). The raw material for their cellulose derivative process is delivered to the factory in form of cellulose sheets. Before entering the process the cellulose sheets are controlled by a viscosity measurement, which functions as a steering parameter for that particular batch. In this data set NIR spectra for 180 cellulose sheets were collected after the sheets had been sent through a grinding process. Hence the NIR spectra were measured on the cellulose raw material in powder form. Data are divided in two parts, one used for modeling and one part for testing.

Format

A list with the following elements:

- nirMN a matrix of 180 samples x 1201 wavelengths in the VIS-NIR region
- viscoVn a vector (length = 180) of viscosity of cellulose powder
- classVn a vector (length = 180) of class membership (1 or 2)

Value

For details see the Format section above.

References

checkW4M

Checking the consistency of an ExpressionSet instance with W4M format

**Description**

Checking the consistency of an ExpressionSet instance with W4M format

**Usage**

```r
checkW4M(eset, ...)
```

**Arguments**

- `eset` An S4 object of class ExpressionSet.
- `...` Currently not used.

**Value**

Invisible TRUE logical in case of success (otherwise generates an error)

**Author(s)**

Etienne Thevenot, <etienne.thevenot@cea.fr>

**Examples**

```r
sacSet <- fromW4M(file.path(path.package("ropls"), "extdata"))
print(checkW4M(sacSet))
```

cost,opls-method

Coefficients method for (O)PLS models

**Description**

Coefficients of the (O)PLS(-DA) regression model

**Usage**

```r
# S4 method for signature 'opls'
coef(object, ...)
```

**Arguments**

- `object` An S4 object of class opls, created by opls function.
- `...` Currently not used.
**cornell**

**Value**

Numeric matrix of coefficients (number of rows equals the number of variables, and the number of columns equals the number of responses)

**Author(s)**

Etienne Thevenot, <etienne.thevenot@cea.fr>

**Examples**

```r
data(sacurine)
attach(sacurine)

sacurine.plsda <- opls(dataMatrix, sampleMetadata[, "gender"])

head(coef(sacurine.plsda))

detach(sacurine)
```

**cornell**

*Octane of various blends of gasoline*

**Description**

Twelve mixture component proportions of the blend are analysed

**Format**

A data frame with the following parameters:

- num mixture number
- x1 proportion of component 1
- x2 proportion of component 2
- x3 proportion of component 3
- x4 proportion of component 4
- x5 proportion of component 5
- x6 proportion of component 6
- x7 proportion of component 7 Note: the 7 variables are correlated since they sum up to 1
- y octane (quantitative variable)

**Value**

Data frame (numeric type only; the first column can be transformed into row names) with 12 rows and 9 columns corresponding to the 'num'ber of the mixture (column 1), the proportion of each of the 7 'x' components within the mixture (columns 2-8), and the octane indice 'y' (column 9). For details see the 'Format' section above.
fitted, opls-method

Source

Tenenhaus (1998), Table 6, page 78.

References


Description

Returns predictions of the (O)PLS(-DA) model on the training dataset

Usage

## S4 method for signature 'opls'
fitted(object, ...) 

Arguments

object

An S4 object of class opls, created by the opls function.

... 

Currently not used.

Value

Predictions (either a vector, factor, or matrix depending on the y response used for training the model)

Author(s)

Etienne Thevenot, <etienne.thevenot@cea.fr>

Examples

data(sacurine) 
attach(sacurine) 
sacurine.plsda <- opls(dataMatrix, sampleMetadata[, "gender"])
fitted(sacurine.plsda)
detach(sacurine)
Description

The relative consumption of 20 food items was compiled for 16 countries. The values range between 0 and 100 percent and a high value corresponds to a high consumption. The dataset contains 3 missing data.

Format

A data frame with the following parameters:

- Country Name of the country
- Gr_CoffeGround Coffee
- Inst_Coffe Instant Coffee
- Tea Tea
- Sweetner Sweetner
- Biscuits Biscuits
- Pa_Soup Powder Soup
- Ti_Soup Tin Soup
- In_Potat Instant Potatoes
- Fro_Fish Frozen Fish
- Fro_Veg Frozen Vegetables
- Apples Apples
- Oranges Oranges
- Ti_Fruit Tin Fruit
- Jam Jam
- Garlic Garlic
- Butter Butter
- Margarine Margarine
- Olive_Oil Olive Oil
- Yoghurt Yoghurt
- Crisp_Brea Crisp Bread

Value

Data frame (numeric type except the first column, which can be transformed into row names) with 16 rows and 21 columns, corresponding to the 'Country' (column 1), followed by the consumption of each of the 20 food items (columns 2-21). For details see the 'Format' section above.

References

Description

Creating a ExpressionSet object from the 3 'dataMatrix.tsv', 'sampleMetadata.tsv' and 'variableMetadata.tsv' tabulated files

Usage

\[
\text{fromW4M(dirC, namePatternC = "", fileTableNamesVc = c("dataMatrix", "sampleMetadata", "variableMetadata"), verboseL = TRUE)}
\]

Arguments

- \text{dirC} 
  Character: directory containing the 3 .tsv files
- \text{namePatternC} 
  Character: optional file name pattern common to all three file names (e.g., when you want to distinguish between two sets of files within the same directory)
- \text{fileTableNamesVc} 
  Vector of characters: if your file names do not contain the standard 'dataMatrix', 'sampleMetadata', and 'variableMetadata' patterns (e.g. if you use 'profile', 'observation', and 'feature' instead), please indicate them here
- \text{verboseL} 
  Logical: should comments be printed?

Value

ExpressionSet instance

Author(s)

Etienne Thevenot, \text{<etienne.thevenot@cea.fr>}

Examples

\[
sacSet \leftarrow \text{fromW4M(file.path(path.package("ropls"), "extdata"))}
\]

getLoadingMN

\section*{getLoadingMN method for PCA(O)PLS(-DA) models}

Description

(Orthogonal) loadings of the PCA/(O)PLS(-DA) model

Usage

\[
 \text{getLoadingMN(object, ...)}
\]

## S4 method for signature 'opls'
\[
\text{getLoadingMN(object, orthoL = FALSE)}
\]
getPcaVarVn

Arguments

object  An S4 object of class opls, created by opls function.
...
Currently not used. (default is FALSE and the predictive loading matrix is returned)
orthoL  Logical: Should the orthogonal loading matrix be returned

Value

Numeric vector with the same length as the number of components

Author(s)

Etienne Thevenot, <etienne.thevenot@cea.fr>

Examples

data(sacurine)
attach(sacurine)
sacurine.plsda <- opls(dataMatrix, 
  sampleMetadata[, "gender"])
getLoadingMN(sacurine.plsda)
detach(sacurine)

getPcaVarVn

getPcaVarVn method for PCA models

Description

Variance of the components (score vectors)

Usage

gPcaVarVn(object, ...)

## S4 method for signature 'opls'
gPcaVarVn(object)

Arguments

object  An S4 object of class opls, created by opls function.
...
Currently not used.

Value

Numeric vector with the same length as the number of components
getScoreMN

Author(s)

Etienne Thevenot, <etienne.thevenot@cea.fr>

Examples

```r
data(sacurine)
attach(sacurine)
sacurine.pca <- opls(dataMatrix)
getPcaVarVn(sacurine.pca)
detach(sacurine)
```

Description

(Orthogonal) scores of the (O)PLS(-DA) model

Usage

```r
getScoreMN(object, ...)
```

## S4 method for signature 'opls'
```r
getScoreMN(object, orthoL = FALSE)
```

Arguments

- `object`: An S4 object of class `opls`, created by `opls` function.
- `...`: Currently not used.
- `orthoL`: Logical: Should the orthogonal score matrix be returned (default is FALSE and the predictive score matrix is returned)

Value

Numeric matrix with a number of rows equal to the number of samples and a number of columns equal to the number of components

Author(s)

Etienne Thevenot, <etienne.thevenot@cea.fr>
Examples

data(sacurine)
attach(sacurine)

sacurine.plsda <- opls(dataMatrix,
    sampleMetadata[, "gender"])

getScoreMN(sacurine.plsda)

detach(sacurine)

getSubsetVi

getSubsetVi method for (O)PLS(-DA) models

Description

Extracts the indices of the samples used for building the model (when a subset argument has been specified)

Usage

getSubsetVi(object, ...)  

# S4 method for signature 'opls'
getSubsetVi(object)

Arguments

object                 An S4 object of class opls, created by opls function.
...                    Currently not used.

Value

Integer vector with the indices of the samples used for training

Author(s)

Etienne Thevenot, <etienne.thevenot@cea.fr>

Examples

data(sacurine)
attach(sacurine)

predictorMN <- dataMatrix
responseFc <- sampleMetadata[, "gender"]

sacurine.plsda <- opls(predictorMN,
    responseFc,
    subset = "odd")
trainVi <- getSubsetVi(sacurine.plsda)

table(responseFc[trainVi], fitted(sacurine.plsda))

detach(sacurine)

---

### Description

Summary of model metrics

### Usage

```r
getSummaryDF(object, ...)
```

### Arguments

- **object**: An S4 object of class `opls`, created by `opls` function.
- **...**: Currently not used.

### Value

Data frame

### Author(s)

Etienne Thevenot, etienne.thevenot@cea.fr

### Examples

```r
data(sacurine)
attach(sacurine)

sacurine.plsda <- opls(dataMatrix, 
    sampleMetadata[, "gender"])

getSummaryDF(sacurine.plsda)
detach(sacurine)
```
**getVipVn**

**getVipVn method for (O)PLS(-DA) models**

**Description**

(Orthogonal) VIP of the (O)PLS(-DA) model

**Usage**

```r
getVipVn(object, ...)  
## S4 method for signature 'opls'
getVipVn(object, orthoL = FALSE)
```

**Arguments**

- **object**: An S4 object of class `opls`, created by `opls` function.
- **...**: Currently not used.
- **orthoL**: Logical: Should the orthogonal VIP be returned (default is FALSE and the predictive VIP is returned)

**Value**

Numeric vector with a length equal to the number of variables and a number of columns equal to the number of components

**Author(s)**

Etienne Thevenot, <etienne.thevenot@cea.fr>

**References**


**Examples**

```r
data(sacurine)
attach(sacurine)

sacurine.plsda <- opls(dataMatrix, 
                      sampleMetadata[, "gender"])

getVipVn(sacurine.plsda)
detach(sacurine)
```
**getWeightMN**

*getWeightMN method for (O)PLS(-DA) models*

**Description**

(Orthogonal) weights of the (O)PLS(-DA) model

**Usage**

`getWeightMN(object, ...)`

```r
## S4 method for signature 'opls'
getWeightMN(object, orthoL = FALSE)
```

**Arguments**

- `object` An S4 object of class `opls`, created by `opls` function.
- `...` Currently not used. (default is FALSE and the predictive weight matrix is returned)
- `orthoL` Logical: Should the orthogonal weight matrix be returned

**Value**

Numeric matrix with a number of rows equal to the number of variables and a number of columns equal to the number of components

**Author(s)**

Etienne Thevenot, <etienne.thevenot@cea.fr>

**Examples**

```r
data(sacurine)
attach(sacurine)

sacurine.plsda <- opls(dataMatrix,
                        sampleMetadata[, "gender"])

getWeightMN(sacurine.plsda)

detach(sacurine)
```
Linnerud Dataset

Description
Three physiological and three exercise variables are measured on twenty middle-aged men in a fitness club.

Format
A data frame with the following parameters:

- num subject number
- weight weight
- waist waist
- pulse pulse
- pullUp pull-up
- squat situp
- jump jump

Value
Data frame (numeric type only; the first column can be transformed into row names) with 20 rows and 7 columns corresponding to the subject’s number (column 1), the 3 physiological variables (columns 2-4), and the 3 exercise variables (columns 5-7). For details see the ‘Format’ section above.

Source
‘mixOmics’ ‘linnerud’ dataset.

References

Description
This example concerns the development of a polymer similar to that used in the plastic covering of mobile phones. The desired profile of the polymer was low warp and high strength. Four constituents (glas, crtp, mica, and amtp) were varied in the polymer formulation by means of a 17 run mixture design. For each new polymer, i.e., each new experiment in the mixture design, 14 responses relating to both warp and strength were measured on the product. The objective of the data analysis was to uncover which combination of factors (the four ingredients) gave polymers with low warp and high strength. The data set contains 10 missing values (NA).
**mark**

**Format**

A data frame with the following parameters:

- num mixture number
- glas glas constituent
- crtp crtp constituent
- mica mica constituent
- amtp amtp constituent
- wrp1 warp response 1
- wrp2 warp response 2
- wrp3 warp response 3
- wrp4 warp response 4
- wrp5 warp response 5
- wrp6 warp response 6
- wrp7 warp response 7
- wrp8 warp response 8
- st1 strength response 1
- st2 strength response 2
- st3 strength response 3
- st4 strength response 4
- st5 strength response 5
- st6 strength response 6

**Value**

Data frame (numeric type only; the first column can be transformed into row names) with 17 rows and 19 columns corresponding to the subject’s number (column 1), the 4 constituent variables (columns 2-5), the 8 warp responses (columns 6-13), and the 6 strength responses (columns 14-19). For details see the ‘Format’ section above.

**References**


---

**mark**

*mark* Dataset

**Description**

Examination marks obtained by French students in Mathematics, Physics, French and English
Format

A data frame with the following parameters:

- nom names of the students
- math marks in mathematics
- phys marks in physics
- fran marks in french
- angl marks in english

Value

Data frame (numeric type except the first column, which can be transformed into row names) with 9 rows and 5 columns, corresponding to the name of the students (column 1), followed by the marks obtained in Maths, Physics, French and English (columns 2-5). For details see the ‘Format’ section above.

Source

'mark' dataset.

References


---

**opls**  PCA, PLS(-DA), and OPLS(-DA)

**Description**

PCA, PLS, and OPLS regression, classification, and cross-validation with the NIPALS algorithm

**Usage**

```r
opls(x, ...)  # S4 method for signature 'ExpressionSet'
```  
```r
opls(x, y = NULL, ...)  # S4 method for signature 'data.frame'
```  
```r
# S4 method for signature 'matrix'
opls(x, y = NULL, predI = NA, orthoI = 0,
     algoC = c("default", "nipals", "svd")[1], crossvalI = 7, log10L = FALSE,
     permI = 20, scaleC = c("none", "center", "pareto", "standard")[4],
     subset = NULL, printL = TRUE, plotL = TRUE, .sinkC = NULL, ...)
```
Arguments

**x**
Numerical data frame or matrix (observations x variables; NAs are allowed); or ExpressionSet object with non empty assayData, for PCA, and phenoData@data, for (O)PLS(-DA), slots

... Currently not used.

**y**
Response to be modelled: Either 1) 'NULL' for PCA (default) or 2) a numerical vector (same length as 'x' row number) for single response (O)PLS, or 3) a numerical matrix (same row number as 'x') for multiple response PLS, 4) a factor (same length as 'x' row number) for (O)PLS-DA, or 5) a character indicating the name of the column of the phenoData@data to be used, when x is an ExpressionSet object. Note that, for convenience, character vectors are also accepted for (O)PLS-DA as well as single column numerical (resp. character) matrix for (O)PLS (respectively (O)PLS-DA). NAs are allowed in numeric responses.

**predI**
Integer: number of components (predictive components in case of PLS and OPLS) to extract; for OPLS, predI is (automatically) set to 1; if set to NA [default], autofit is performed: a maximum of 10 components are extracted until (i) PCA case: the variance is less than the mean variance of all components (note that this rule requires all components to be computed and can be quite time-consuming for large datasets) or (ii) PLS case: either R2Y of the component is < 0.01 (N4 rule) or Q2Y is < 0 (for more than 100 observations) or 0.05 otherwise (R1 rule)

**orthoI**
Integer: number of orthogonal components (for OPLS only); when set to 0 [default], PLS will be performed; otherwise OPLS will be performed; when set to NA, OPLS is performed and the number of orthogonal components is automatically computed by using the cross-validation (with a maximum of 9 orthogonal components).

**algoC**
Default algorithm is 'svd' for PCA (in case of no missing values in 'x'; 'nipals’ otherwise) and 'nipals' for PLS and OPLS; when asking to use ‘svd’ for PCA on an 'x' matrix containing missing values, NAs are set to half the minimum of non-missing values and a warning is generated

**crossvalI**
Integer: number of cross-validation segments (default is 7); The number of samples (rows of 'x') must be at least >= crossvalI

**log10L**
Should the 'x' matrix be log10 transformed? Zeros are set to 1 prior to transformation

**permI**
Integer: number of random permutations of response labels to estimate R2Y and Q2Y significance by permutation testing [default is 20 for single response models (without train/test partition), and 0 otherwise]

**scaleC**
Character: either no centering nor scaling ('none'), mean-centering only ('center'), mean-centering and pareto scaling ('pareto'), or mean-centering and unit variance scaling ('standard') [default]

**subset**
Integer vector: indices of the observations to be used for training (in a classification scheme); use NULL [default] for no partition of the dataset; use 'odd' for a partition of the dataset in two equal sizes (with respect to the classes proportions)

**printL**
Logical: Should informations regarding the data set and the model be printed? [default = TRUE]

**plotL**
Logical: Should the 'summary' plot be displayed? [default = TRUE]

**.sinkC**
Character: Name of the file for R output diversion [default = NULL: no diversion]; Diversion of messages is required for the integration into Galaxy
An S4 object of class 'opls' containing the following slots:

- **typeC** Character: model type (PCA, PLS, PLS-DA, OPLS, or OPLS-DA)
- **descriptionMC** Character matrix: Description of the data set (number of samples, variables, etc.)
- **modelDF** Data frame with the model overview (number of components, R2X, R2X(cum), R2Y, R2Y(cum), Q2, Q2(cum), significance, iterations)
- **summaryDF** Data frame with the model summary (cumulated R2X, R2Y and Q2); RMSEE is the square root of the mean error between the actual and the predicted responses
- **subsetVi** Integer vector: Indices of observations in the training data set
- **pcaVarVn** PCA: Numerical vector of variances of length: predI
- **vipVn** PLS(-DA): Numerical vector of Variable Importance in Projection; OPLS(-DA): Numerical vector of Variable Importance for Prediction (VIP4,p from Galindo-Prieto et al, 2014)
- **orthoVipVn** OPLS(-DA): Numerical vector of Variable Importance for Orthogonal Modeling (VIP4,o from Galindo-Prieto et al, 2014)
- **xMeanVn** Numerical vector: variable means of the 'x' matrix
- **xSdVn** Numerical vector: variable standard deviations of the 'x' matrix
- **yMeanVn** (O)PLS: Numerical vector: variable means of the 'y' response (transformed into a dummy matrix in case it is of 'character' mode initially)
- **ySdVn** (O)PLS: Numerical vector: variable standard deviations of the 'y' response (transformed into a dummy matrix in case it is of 'character' mode initially)
- **xZeroVarVi** Numerical vector: indices of variables with variance < 2.22e-16 which were excluded from 'x' before building the model
- **scoreMN** Numerical matrix of x scores (T; dimensions: nrow(x) x predI) \( X = TP' + E \); \( Y = TC' + F \)
- **loadingMN** Numerical matrix of x loadings (P; dimensions: ncol(x) x predI) \( X = TP' + E \)
- **weightMN** (O)PLS: Numerical matrix of x weights (W; same dimensions as loadingMN)
- **orthoScoreMN** OPLS: Numerical matrix of orthogonal scores (Tortho; dimensions: nrow(x) x number of orthogonal components)
- **orthoLoadingMN** OPLS: Numerical matrix of orthogonal loadings (Portho; dimensions: ncol(x) x number of orthogonal components)
- **orthoWeightMN** OPLS: Numerical matrix of orthogonal weights (same dimensions as orthoLoadingMN)
- **cMN** (O)PLS: Numerical matrix of Y weights (C; dimensions: number of responses or number of classes in case of qualitative response) x number of predictive components; \( Y = TC' + F \)
- **coMN** (O)PLS: Numerical matrix of Y orthogonal weights; dimensions: number of responses or number of classes in case of qualitative response with more than 2 classes x number of orthogonal components
- **uMN** (O)PLS: Numerical matrix of Y scores (U; same dimensions as scoreMN); \( Y = UC' + G \)
- **weightStarMN** Numerical matrix of projections (W*; same dimensions as loadingMN); whereas columns of weightMN are derived from successively deflated matrices, columns of weightStarMN relate to the original 'x' matrix: \( T = XW*; W* = W(P'W)inv \)
- **suppLs** List of additional objects to be used internally by the 'print', 'plot', and 'predict' methods
Author(s)
Etienne Thevenot, <etienne.thevenot@cea.fr>

References

Examples

#### PCA

data(foods) ## see Eriksson et al. (2001); presence of 3 missing values (NA)
head(foods)
foodMN <- as.matrix(foods[, colnames(foods) != "Country"])
rownames(foodMN) <- foods[, "Country"]
head(foodMN)
foo.pca <- opls(foodMN)

#### PLS with a single response

data(cornell) ## see Tenenhaus, 1998
head(cornell)
cornell.pls <- opls(as.matrix(cornell[, grep("x", colnames(cornell))]),
cornell[, "y"])

#### PLS with multiple (quantitative) responses

data(lowarp) ## see Eriksson et al. (2001); presence of NAs
head(lowarp)
lowarp.pls <- opls(as.matrix(lowarp[, c("glas", "crtp", "mica", "amtp")]),
as.matrix(lowarp[, grepl("^wrp", colnames(lowarp)) |
grepl("^st", colnames(lowarp))]))

#### PLS-DA

data(sacurine)
attach(sacurine)
sacurine.plsda <- opls(dataMatrix, sampleMetadata[, "gender"])

#### OPLS-DA
sacurine.oplsda <- opls(dataMatrix, sampleMetadata[, "gender"], predI = 1, orthoI = NA)
detach(sacurine)
An S4 class to store PCA and (O)PLS(-DA) models: Objects can be created by calls of the form `new("opls", ...)` or by calling the `opls` function.

### Slots

- **typeC**: character: model type (PCA, PLS, PLS-DA, OPLS, or OPLS-DA)
- **descriptionMC**: character matrix: Description of the data set (number of samples, variables, etc.)
- **modelDF**: data frame with the model overview (number of components, $R^2_X$, $R^2_X(cum)$, $R^2_Y$, $R^2_Y(cum)$, $Q^2$, $Q^2(cum)$, significance, iterations)
- **summaryDF**: data frame with the model summary (cumulated $R^2_X$, $R^2_Y$ and $Q^2$); RMSEE is the square root of the mean error between the actual and the predicted responses
- **subsetVi**: Integer vector: Indices of observations in the training data set
- **pcaVarVn**: PCA: Numerical vector of variances of length: predI
- **vipVn**: PLS(-DA): Numerical vector of Variable Importance in Projection; OPLS(-DA): Numerical vector of Variable Importance for Prediction (VIP4,p from Galindo-Prieto et al, 2014)
- **orthoVipVn**: OPLS(-DA): Numerical vector of Variable Importance for Orthogonal Modeling (VIP4,o from Galindo-Prieto et al, 2014)
- **coefficientMN**: (O)PLS(-DA): Numerical matrix of regression coefficients ($B$; dimensions: $n_{col(x)} \times n_{row(x)}$; $B = W \times C'$ and $Y = XB + F$)
- **xMeanVn**: Numerical vector: variable means of the 'x' matrix
- **xSdVn**: Numerical vector: variable standard deviations of the 'x' matrix
- **yMeanVn**: (O)PLS: Numerical vector: variable means of the 'y' response (transformed into a dummy matrix in case it is of 'character' mode initially)
- **ySdVn**: (O)PLS: Numerical vector: variable standard deviations of the 'y' response (transformed into a dummy matrix in case it is of 'character' mode initially)
- **xZeroVarVi**: Numerical vector: indices of variables with variance < 2.22e-16 which were excluded from 'x' before building the model
- **scoreMN**: Numerical matrix of x scores (T; dimensions: $n_{row(x)} \times predI$) $X = TP' + E$; $Y = TC' + F$
- **loadingMN**: Numerical matrix of x loadings (P; dimensions: $n_{col(x)} \times predI$) $X = TP' + E$
- **weightMN**: (O)PLS: Numerical matrix of x weights (W; same dimensions as loadingMN)
- **orthoScoreMN**: OPLS: Numerical matrix of orthogonal scores (Tortho; dimensions: $n_{row(x)} \times number of orthogonal components$)
- **orthoLoadingMN**: OPLS: Numerical matrix of orthogonal loadings (Portho; dimensions: $n_{col(x)} \times number of orthogonal components$)
- **orthoWeightMN**: OPLS: Numerical matrix of orthogonal weights (same dimensions as orthoLoadingMN)
- **cMN**: (O)PLS: Numerical matrix of Y weights (C; dimensions: number of responses or number of classes in case of qualitative response with more than 2 classes x number of predictive components; $Y = TC' + F$)
coMN (O)PLS: Numerical matrix of Y orthogonal weights; dimensions: number of responses or number of classes in case of qualitative response with more than 2 classes x number of orthogonal components

uMN (O)PLS: Numerical matrix of Y scores (U; same dimensions as scoreMN); Y = UC' + G

weightStarMN Numerical matrix of projections (W*; same dimensions as loadingMN); whereas columns of weightMN are derived from successively deflated matrices, columns of weight-StarMN relate to the original 'x' matrix: T = XW*; W* = (W'W)inv

suppLs List of additional objects to be used internally by the 'print', 'plot', and 'predict' methods

Objects from the Class

Objects can be created by calls of the form new("opls", ...) or by calling the opls function

Author(s)

Etienne Thevenot, <etienne.thevenot@cea.fr>

See Also

opls

Examples

#### PCA

data(foods) ## see Eriksson et al. (2001); presence of 3 missing values (NA)
head(foods)
foodMN <- as.matrix(foods[, colnames(foods) != "Country"])
rownames(foodMN) <- foods[, "Country"]
head(foodMN)
foo.pca <- opls(foodMN)

#### PLS with a single response

data(cornell) ## see Tenenhaus, 1998
head(cornell)
cornell.pls <- opls(as.matrix(cornell[, grep("x", colnames(cornell)))],
                    cornell[, "y"])

## Complementary graphics

plot(cornell.pls, typeVc = c("outlier", "predict-train", "xy-score", "xy-weight"))

#### PLS with multiple (quantitative) responses

data(lowarp) ## see Eriksson et al. (2001); presence of NAs
head(lowarp)
lowarp.pls <- opls(as.matrix(lowarp[, c("glas", "crtp", "mica", "amtp")]),
                   as.matrix(lowarp[, grep("warp", colnames(lowarp)) |
                                   grep("st", colnames(lowarp))]))

#### PLS-DA
data(sacurine)
attach(sacurine)
sacurine.plsda <- opls(dataMatrix, sampleMetadata[, "gender"])

### OPLS-DA
sacurine.oplsda <- opls(dataMatrix, sampleMetadata[, "gender"], predI = 1, orthoI = NA)
detach(sacurine)

---

**plot,opls-method**

*Plot Method for (O)PLS(-DA)*

**Description**

This function plots values based upon a model trained by opls.

**Usage**

```r
# S4 method for signature 'opls'
plot(x, y, typeVc = c("correlation", "outlier", "overview", "permutation", "predict-train", "predict-test", "summary", "x-loading", "x-score", "x-variance", "xy-score", "xy-weight")[7], parAsColFcVn = NA, parCexN = 0.8, parCompVi = c(1, 2), parDevNewL = TRUE, parEllipsesL = NA, parLabVc = NA, parTitleL = TRUE, file.pdfC = NULL, .sinkC = NULL, ...)```

**Arguments**

- `x` An S4 object of class opls, created by the opls function.
- `y` Currently not used.
- `typeVc` Character vector: the following plots are available: 'correlation': Variable correlations with the components, 'outlier': Observation diagnostics (score and orthogonal distances), 'overview': Model overview showing R2Ycum and Q2cum (or 'Variance explained' for PCA), 'permutation': Scatterplot of R2Y and Q2Y actual and simulated models after random permutation of response values; 'predict-train' and 'predict-test': Predicted vs Actual Y for reference and test sets (only if Y has a single column), 'summary' [default]: 4-plot summary showing permutation, overview, outlier, and x-score together, 'x-variance': Spread of raw variables correspondingly with min, median, and max variances, 'x-loading': X-loadings (the 6 of variables most contributing to loadings are colored in red to facilitate interpretation), 'x-score': X-Scores, 'xy-score': XY-Scores, 'xy-weight': XY-Weights
- `parAsColFcVn` Optional factor character or numeric vector to be converted into colors for the score plot; default is NA [ie colors will be converted from `y` in case of (O)PLS(-DA) or will be 'black' for PCA]
- `parCexN` Numeric: amount by which plotting text should be magnified relative to the default
- `parCompVi` Integer vector of length 2: indices of the two components to be displayed on the score plot (first two components by default)
plot.opls-method

parDevNewL Should the graphics be displayed in a new window [default]; If FALSE, parLayL must be set to FALSE also

parEllipsesL Should the Mahalanobis ellipses be drawn? If `NA' [default], ellipses are drawn when either a character parAsColVcn is provided (PCA case), or when `y' is a character factor ((O)PLS-DA cases).

parLabVc Optional character vector for the labels of observations on the plot; default is NA [ie row names of `x', if available, or indices of `x', otherwise, will be used]

parTitleL Should the titles of the plots be printed on the graphics (default = TRUE); It may be convenient to set this argument to FALSE when the user wishes to add specific titles a posteriori

file.pdfC Figure filename (e.g. in case of batch mode) ending with `.pdf'; for multiple graphics, set parLayL to TRUE; default is NULL (no saving; displaying instead)

.sinkC Character: Name of the file for R output diversion [default = NULL: no diversion]; Diversion of messages is required for the integration into Galaxy

... Currently not used.

Author(s)

Etienne Thevenot, <etienne.thevenot@cea.fr>

Examples

data(sacurine)
attach(sacurine)

for(typeC in c("correlation", "outlier", "overview", "permutation", "predict-train", "predict-test", "summary", "x-loading", "x-score", "x-variance", "xy-score", "xy-weight")) {

print(typeC)

if(grepl("predict", typeC))
  subset <- "odd"
else
  subset <- NULL

opLs <- opls(dataMatrix, sampleMetadata[, "gender"],
  predI = ifelse(typeC != "xy-weight", 1, 2),
  orthoI = ifelse(typeC != "xy-weight", 1, 0),
  permI = ifelse(typeC == "permutation", 10, 0),
  subset = subset,
  printL = FALSE, plotL = FALSE)

plot(opLs, typeVc = typeC)
}

detach(sacurine)
predict, opls-method  
*Predict method for (O)PLS models*

**Description**

Returns predictions of the (O)PLS(-DA) model on a new dataset.

**Usage**

```r
## S4 method for signature 'opls'
predict(object, newdata, ...)
```

**Arguments**

- `object`: An S4 object of class `opls`, created by `opls` function.
- `newdata`: Either a data frame or a matrix, containing numeric columns only, with the same number of columns (variables) as the `x` used for model training with `opls`.
- `...`: Currently not used.

**Value**

Predictions (either a vector, factor, or matrix depending on the y response used for training the model)

**Author(s)**

Etienne Thevenot, <etienne.thevenot@cea.fr>

**Examples**

```r
data(sacurine)
attach(sacurine)

predictorMN <- dataMatrix
responseFc <- sampleMetadata[, "gender"]

sacurine.plsda <- opls(predictorMN, responseFc, subset = "odd")

trainVi <- getSubsetVi(sacurine.plsda)
table(responseFc[trainVi], fitted(sacurine.plsda))
table(responseFc[-trainVi],
       predict(sacurine.plsda, predictorMN[-trainVi, ]))

detach(sacurine)
```
### Description
Displays information about the dataset and the model.

### Usage
```r
## S4 method for signature 'opls'
print(x, ...
```

### Arguments
- `x` An S4 object of class `opls`, created by the `opls` function.
- `...` Currently not used.

### Value
Invisible.

### Author(s)
Etienne Thevenot, <etienne.thevenot@cea.fr>

### Examples
```r
data(sacurine)
attach(sacurine)
sacurine.plsda <- opls(dataMatrix, sampleMetadata[, "gender"])
p
print(sacurine.plsda)
detach(sacurine)
```

---

### Description
Returns the residuals from the (O)PLS(-DA) regression models

### Usage
```r
## S4 method for signature 'opls'
residuals(object, ...
```
Arguments

object  An S4 object of class opls, created by opls function.

Value

Numeric matrix or vector (same dimensions as the modeled y response); if y is a character vector or a factor (in case of classification), the residuals equal 0 (predicted class identical to the true class) or 1 (prediction error).

Author(s)

Etienne Thevenot, <etienne.thevenot@cea.fr>

Examples

data(sacurine)
attach(sacurine)

sacurine.pls <- opls(dataMatrix,
                     sampleMetadata[, "age"])

head(residuals(sacurine.pls))
detach(sacurine)

sacurine  Analysis of the human adult urinary metabolome variations with age, body mass index and gender

Description

Urine samples from 183 human adults were analyzed by liquid chromatography coupled to high-resolution mass spectrometry (LTQ Orbitrap) in the negative ionization mode. A total of 109 metabolites were identified or annotated at the MSI level 1 or 2. After retention time alignment with XCMS, peaks were integrated with Quan Browser. After signal drift and batch effect correction of intensities, each urine profile was normalized to the osmolality of the sample. Finally, the data were log10 transformed.

Format

A list with the following elements:

- dataMatrix a 183 samples x 109 variables matrix of numeric type corresponding to the intensity profiles (values have been log10-transformed)
- sampleMetadata a 183 x 3 data frame, with the volunteers’ age (’age’, numeric), body mass index (’bmi’, numeric), and gender (’gender’, factor)
- variableMetadata a 109 x 3 data frame, with the metabolites’ MSI identification level (’msiLevel’: either 1 or 2), HMDB ID when available (’hmdb’, character), chemical class according to the ’super class’ taxonomy of HMDB (’chemicalClass’, character)
show.opls-method

Value
List containing the 'dataMatrix' matrix (numeric) of data (samples as rows, variables as columns),
the 'sampleMetadata' data frame of sample metadata, and the variableMetadata data frame of vari-
able metadata. Row names of 'dataMatrix' and 'sampleMetadata' are identical. Column names of
'dataMatrix' are identical to row names of 'variableMetadata'. For details see the 'Format' section
above.

References
inary metabolome variations with age, body mass index and gender by implementing a comprehen-
sive workflow for univariate and OPLS statistical analyses. Journal of Proteome Research, DOI:
10.1021/acs.jproteome.5b00354

show,opls-method

Show method for 'opls' objects

Description
Displays information about the dataset and the model.

Usage
## S4 method for signature 'opls'
show(object)

Arguments

object
An S4 object of class opls, created by the opls function.

Value
Invisible.

Author(s)
Philippe Rinaudo and Etienne Thevenot (CEA)

Examples

data(sacurine)
attach(sacurine)
sacurine.plsda <- opls(dataMatrix, sampleMetadata[, "gender"])

show(sacurine.plsda)
detach(sacurine)
**strF**

*Printed summary of an R object*

**Description**

Displays the class, mode, size and first...last values of the object

**Usage**

```r
cstrF(inpMF, borderN = 2, bigMarkC = ",")
```

**Arguments**

- `inpMF`: Input matrix, dataframe or vector
- `borderN`: Number of border (first and last) rows and columns to display
- `bigMarkC`: Big mark separator for summary results

**Value**

This function has no output.

**Author(s)**

Etienne Thevenot (CEA)

**See Also**

`str`

**Examples**

```r
data(sacurine)
cstrF(sacurine[["dataMatrix"]])
cstrF(sacurine[["sampleMetadata"]])
```

---

**tested**

*Tested method for (O)PLS models*

**Description**

Returns predictions of the (O)PLS(-DA) model on the out of the box samples (when a ‘subset’ of samples has been selected when training the model)

**Usage**

```r
tested(object, ...)
```

```r
# S4 method for signature 'opls'
tested(object)
```
Arguments

object An S4 object of class opls, created by opls function.
... Currently not used.

Value

Predictions (either a vector, factor, or matrix depending on the y response used for training the model)

Author(s)

Etienne Thevenot, <etienne.thevenot@cea.fr>

Examples

data(sacurine)
attach(sacurine)

testedorMN <- dataMatrix
responseFc <- sampleMetadata[, "gender"]

sacurine.plsda <- opls(testedorMN,
 alternatively
responseFc,
 subset = "odd")

trainVi <- getSubsetVi(sacurine.plsda)

table(responseFc[trainVi], fitted(sacurine.plsda))
detach(sacurine)

toW4M

Exporting ExpressionSet instance into 3 tabulated files.

Description

The 3 .tsv files are written with the indicated file prefix, and '_dataMatrix.tsv', '_sampleMetadata.tsv', and '_variableMetadata.tsv' suffices, respectively. Note that the dataMatrix is transposed before export (e.g., the samples are written column wise in the 'dataMatrix.tsv' exported file).

Usage

toW4M(eset, ...)

## S4 method for signature 'ExpressionSet'
toW4M(eset, filePrefixC = paste0(getwd(), "/out_"),
 alternatively
verboseL = TRUE, ...)
Arguments

- **eset**: An S4 object of class `ExpressionSet` function.
- **...**: Currently not used.
- **filePrefixC**: Character: common prefix (including repository full path) of the three file names: for example, the `c:/mydata/setname` value will result in writing the `c:/mydata/setname_dataMatrix.tsv`, `c:/mydata/setname_sampleMetadata.tsv`, and `c:/mydata/setname_variableMetadata.tsv` files.
- **verboseL**: Logical: should comments be printed?

Value

No object returned.

Author(s)

Etienne Thevenot, <etienne.thevenot@cea.fr>

Examples

```r
sacSet <- fromW4M(file.path(path.package("ropls"), "extdata"))
toW4M(sacSet)
```
Index

*Topic **datasets**
  - aminoacids, 3
  - cellulose, 4
  - cornell, 6
  - foods, 8
  - linnerud, 16
  - lowarp, 16
  - mark, 17
  - sacurine, 28

*Topic **package**
  - ropls-package, 2

aminoacids, 3

cellulose, 4
checkW4M, 5
checkW4M, ExpressionSet-method
  (checkW4M), 5
coeff, opls-method, 5
coeff.opls(coeff, opls-method), 5
cornell, 6

fitted, opls-method, 7
fitted.opls(fitted, opls-method), 7
foods, 8
fromW4M, 9

getLoadingMN, 9
getLoadingMN, opls-method
  (getLoadingMN), 9
getPcaVarVn, 10
getPcaVarVn(opls-method(getPcaVarVn), 10
getScoreMN, 11
getScoreMN, opls-method(getScoreMN), 11
getSubsetVi, 12
getSubsetVi(opls-method(getSubsetVi), 12
getSummaryDF, 13
getSummaryDF(opls-method(getSummaryDF), 13
getVipVn, 14
getVipVn(opls-method(getVipVn), 14
getWeightMN, 15

getWeightMN(opls-method(getWeightMN), 15

linnerud, 16
lowarp, 16
mark, 17

opls, 18, 23
opls(data.frame-method(opls), 18
opls(ExpressionSet-method(opls), 18
opls(matrix-method(opls), 18
opls-class, 22

plot, opls-method, 24
plot.opls(plot, opls-method), 24
predict, opls-method, 26
predict.opls(predict, opls-method), 26
print, opls-method, 27
print.opls(print, opls-method), 27

residuals, opls-method, 27
residuals.opls(residuals, opls-method), 27

ropls(ropls-package), 2
ropls-package, 2

sacurine, 28
show, opls-method, 29
show.opls(show, opls-method), 29
str, 30
strF, 30

tested, 30
tested(opls-method(tested), 30
toW4M, 31
toW4M, ExpressionSet-method(toW4M), 31