Package 'ropls'

May 30, 2024

Type Package

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

Version 1.36.0

Date 2023-12-06

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

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).

Depends R (>= 3.5.0)

Imports Biobase, ggplot2, graphics, grDevices, methods, plotly, stats, MultiAssayExperiment, MultiDataSet, SummarizedExperiment, utils

Suggests BiocGenerics, BiocStyle, knitr, multtest, omicade4, rmarkdown, testthat

VignetteBuilder knitr

License CeCILL

Encoding UTF-8

LazyLoad yes

URL https://doi.org/10.1021/acs.jproteome.5b00354

NeedsCompilation no

RoxygenNote 7.2.3

git_url https://git.bioconductor.org/packages/ropls

git_branch RELEASE_3_19

git_last_commit ca86d7b

git_last_commit_date 2024-04-30

Repository Bioconductor 3.19

Date/Publication 2024-05-29

Author Etienne A. Thevenot [aut, cre] (<https://orcid.org/0000-0003-1019-4577>)

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

Contents

`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). Input formats include matrix, data frame, SummarizedExperiment, MultiAssayExperiment, ExpressionSet, MultiDataSet. The package can be accessed via a user interface on the Workflow4Metabolomics online resource built upon the Galaxy environment.

Author(s)

Etienne A. Thevenot (CEA)

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

`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 contaning information about amino acids. For details see the 'Format' section above.

Source

'aminoacids' dataset.

References

Wold et al. (2001). PLS-regression: a basic tool of chemometrics. Chemometrics and Intelligent Laboratory Systems. 58:109-130.

`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

Multivariate calibration using spectral data. Simca tutorial. Umetrics.

`checkW4M` *Checking the consistency of a SummarizedExperiment or ExpressionSet instance with W4M format*

Description

Checking the consistency of a SummarizedExperiment or ExpressionSet instance with W4M format

Usage

```
checkW4M(x)

## S4 method for signature 'SummarizedExperiment'
checkW4M(x)

## S4 method for signature 'ExpressionSet'
checkW4M(x)
```

Arguments

x	An S4 object of class SummarizedExperiment or ExpressionSet

Value

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

Author(s)

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

Examples

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

---

coef,opls-method *Coefficients method for (O)PLS models*

Description

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

Usage

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

Arguments

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

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

```
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.

Source

Tenenhaus (1998), Table 6, page 78.

References

Tenenhaus (1998). La regression PLS: theorie et pratique. Paris: Editions Technip.

fitted,opls-method *Fitted method for 'opls' objects*

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.

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)
```

foods *Food consumption patterns accross European countries (FOODS)*

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

Eriksson et al. (2006). Multi- and Megarvariate Data Analysis. Umetrics Academy. pp.10, 33, 48.

fromW4M *fromW4M*

Description

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

Usage

```
fromW4M(
  dirC,
  namePatternC = "",
  fileTableNamesVc = c("dataMatrix", "sampleMetadata", "variableMetadata"),
  outputC = c("exp", "set")[2],
  verboseL = TRUE
)
```

Arguments

dirC	Character: directory containing the 3 .tsv files
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)
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
outputC	character(1): either 'exp' for SummarizedExperiment (default) or 'set' for ExpressionSet output format
verboseL	Logical: should comments be printed?

Value

ExpressionSet instance

Author(s)

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

Examples

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

getEset *getEset method*

Description

Extracts the complemented ExpressionSet when opls has been applied to an ExpressionSet

Usage

```
getEset(object)

## S4 method for signature 'opls'
getEset(object)
```

Arguments

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

Value

An S4 object of class ExpressionSet which contains the dataMatrix (t(exprs(eset))), and the sampleMetadata (pData(eset)) and variableMetadata (fData(eset)) with the additional columns containing the scores, predictions, loadings, VIP, coefficients etc.

Author(s)

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

Examples

```
data(sacurine)
sacSet <- sacurine[["eset"]]
sacPlsda <- opls(sacSet, "gender")
sacSet <- getEset(sacPlsda)
head(Biobase::pData(sacSet))
head(Biobase::fData(sacSet))
```

getLoadingMN *getLoadingMN method for PCA/(O)PLS(-DA) models*

Description

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

Usage

```
getLoadingMN(object, orthoL = FALSE)

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

Arguments

object	An S4 object of class opls, created by opls function.
orthoL	Logical: Should the orthogonal loading matrix be returned (default is FALSE and the predictive loading matrix is 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

```
data(sacurine)
attach(sacurine)

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

getLoadingMN(sacurine.plsda)

detach(sacurine)
```

---

getMset *getMset method*

Description

Extracts the complemented MultiDataSet when opls has been applied to a MultiDataSet

Usage

```
getMset(object)

## S4 method for signature 'oplsMultiDataSet'
getMset(object)
```

Arguments

object	An S4 object of class oplsMultiDataSet, created by opls function applied to a MultiDataSet

Value

An S4 object of class MultiDataSet.

Examples

```
data(NCI60)
nci.mds <- NCI60[["mds"]]
# Restricting to the 'agilent' and 'hgu95' datasets
nci.mds <- nci.mds[, c("agilent", "hgu95")]
# Restricting to the 'ME' and 'LE' cancer types
sampleNamesVc <- Biobase::sampleNames(nci.mds[["agilent"]])
cancerTypeVc <- Biobase::pData(nci.mds[["agilent"]])[, "cancer"]
nci.mds <- nci.mds[sampleNamesVc[cancerTypeVc %in% c("ME", "LE")], ]
# Principal Component Analysis of each data set
nci.pca <- opls(nci.mds)
# Getting the MultiDataSet with additional info. in pData and fData
nci.mds <- getMset(nci.pca)
```

---

getOpls *Getting the models from a SummarizedExperiment or a MultiAssayExperiment object*

Description

The models are extracted as a list

Usage

```
getOpls(object)

## S4 method for signature 'SummarizedExperiment'
getOpls(object)

## S4 method for signature 'MultiAssayExperiment'
getOpls(object)
```

Arguments

object	An S4 object of class SummarizedExperiment or MultiAssayExperiment, once processed by the opls method

Value

List of opls models contained in the SummarizedExperiment object(s)

Author(s)

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

Examples

```
# Getting the sacurine data set as a SummarizedExperiment
data(sacurine)
sac.se <- sacurine[["se"]]

# Building the PLS-DA model
sac.se <- opls(sac.se, "gender")

# Getting the models
sac_opls.ls <- getOpls(sac.se)
names(sac_opls.ls)

# Plotting the score plot from the PLS-DA model of the gender response
plot(sac_opls.ls[["gender_PLSDA"]], typeVc = "x-score")
```

---

getPcaVarVn *getPcaVarVn method for PCA models*

Description

Variance of the components (score vectors)

Usage

```
getPcaVarVn(object)

## S4 method for signature 'opls'
getPcaVarVn(object)
```

Arguments

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

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.pca <- opls(dataMatrix)

getPcaVarVn(sacurine.pca)

detach(sacurine)
```

---

getScoreMN *getScoreMN method for PCA/(O)PLS(-DA) models*

Description

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

Usage

```
getScoreMN(object, orthoL = FALSE)

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

Arguments

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

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)
```

getSummaryDF *getSummaryDF method for PCA/(O)PLS models*

Description

Summary of model metrics

Usage

```
getSummaryDF(object)

## S4 method for signature 'opls'
getSummaryDF(object)
```

Arguments

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

Value

Data frame

Author(s)

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

Examples

```
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

```
getVipVn(object, orthoL = FALSE)

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

Arguments

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

Galindo-Prieto B., Eriksson L. and Trygg J. (2014). Variable influence on projection (VIP) for orthogonal projections to latent structures (OPLS). Journal of Chemometrics 28, 623-632.

Examples

```
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, orthoL = FALSE)

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

Arguments

object	An S4 object of class opls, created by opls function.
orthoL	Logical: Should the orthogonal weight matrix be returned? (default is FALSE)

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

```
data(sacurine)
attach(sacurine)

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

getWeightMN(sacurine.plsda)

detach(sacurine)
```

gg_scoreplot *PCA and (O)PLS(-DA) score plots*

Description

Score plot visualization for PCA and (O)PLS(-DA) models in either ggplot or ggplotly formats

Usage

```
gg_scoreplot(
  x,
  model.c = "",
  components.vi = c(1, 2),
  label.c = c("", "sample_names")[2],
  color.c = "",
  title.c = "",
  palette.c = "Set1",
  legend.c = "right",
  ellipse.l = TRUE,
  plotly.l = FALSE,
  info.vc = "sample_names",
 size.ls = list(axis_lab.i = 16, axis_text.i = 14, point.i = 3, label.i = 5, title.i =
    20, legend_title.i = 15, legend_text.i = 15)
)

## S4 method for signature 'SummarizedExperiment'
gg_scoreplot(
  x,
  model.c = "",
  components.vi = c(1, 2),
  label.c = c("", "sample_names")[2],
  color.c = "",
  title.c = "",
  palette.c = "Set1",
  legend.c = "right",
  ellipse.l = TRUE,
  plotly.l = FALSE,
  info.vc = "sample_names",
 size.ls = list(axis_lab.i = 16, axis_text.i = 14, point.i = 3, label.i = 5, title.i =
    20, legend_title.i = 15, legend_text.i = 15)
)

## S4 method for signature 'opls'
gg_scoreplot(
  x,
  model.c = "",
  components.vi = c(1, 2),
```

```
  label.c = c("", "sample_names")[2],
  color.c = "",
  title.c = "",
  palette.c = "Set1",
  legend.c = "right",
  ellipse.l = TRUE,
  plotly.l = FALSE,
  info.vc = "sample_names",
 size.ls = list(axis_lab.i = 16, axis_text.i = 14, point.i = 3, label.i = 5, title.i =
    20, legend_title.i = 15, legend_text.i = 15)
)
```

Arguments

x	An S4 object of class SummarizedExperiment (resp. opls) generated by the 'ropls::opls' modeling applied to a SummarizedExperiment (resp. an ExpressionSet)
model.c	character(1): name of the model to be plotted; use 'names(ropls::getOpls(se))' to see the available models in the se object
components.vi	integer(2): number of the components to display as x and y axis
label.c	character(1): name of the colData (resp. pData) column to be used for the labels
color.c	character(1): name of the colData (resp. pData) column to be used for the colors
title.c	character(1): plot title
palette.c	character(1): name of the RColorBrewer palette (for qualitative factor)
legend.c	character(1): position of the legend (either 'bottom', 'left', 'top' or 'right' [default])
ellipse.l	logical(1): should ellipses be drawn (for qualitative factor)
plotly.l	logical(1): should the ggplot be converted to an interactive plotly (default: FALSE)
info.vc	character(): names of the colData (resp. pData) columns to be used for the plotly info; the default 'sample_names' will return the sample names as the plotly info
size.ls	list: sizes for axis labels (default: 16), axis text (default: 14), points (default: 3), labels (default = 5), title (default = 20), legend title (default: 15), legend text (default: 15)

Value

invisible ggplot2 (or ggplotly) object

Examples

```
# loading the 'sacurine' dataset from the 'ropls' package
data(sacurine, package = "ropls")
# SummarizedExperiment
sac.se <- sacurine[["se"]]
## computing the PCA
sac.se <- ropls::opls(sac.se)
## score plot
```

```
gg_scoreplot(sac.se, "PCA")
gg_scoreplot(sac.se, "PCA", color.c = "age")
gg_scoreplot(sac.se, "PCA", color.c = "gender", plotly.l = TRUE, info.vc = "all")
## PLS-DA modeling
sac.se <- ropls::opls(sac.se, "gender")
gg_scoreplot(sac.se, "gender_PLSDA", color.c = "gender")
gg_scoreplot(sac.se, "gender_PLSDA", color.c = "gender", plotly.l = TRUE)
## OPLS-DA modeling
sac.se <- ropls::opls(sac.se, "gender", predI = 1, orthoI = NA)
gg_scoreplot(sac.se, "gender_OPLSDA", color.c = "gender")
gg_scoreplot(sac.se, "gender_OPLSDA", color.c = "gender", plotly.l = TRUE, info.vc = "all")
# ExpressionSet
sacurine.eset <- sacurine[["eset"]]
## PCA
sacurine.pca <- ropls::opls(sacurine.eset)
## score plot (model.c does not need to be specified here since 'opls' objects contain only one model)
gg_scoreplot(sacurine.pca)
gg_scoreplot(sacurine.pca, color.c = "age")
```

`linnerud` *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 'num'ber (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

Tenenhaus (1998). La regression PLS: theorie et pratique. Paris: Editions Technip.

`lowarp` *A multi response optimization data set (LOWARP)*

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).

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 'num'ber (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

Eriksson et al. (2006). Multi- and Megarvariate Data Analysis. Umetrics Academy. pp.16, 77, 209.

`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

Baccini (2010). Statistique Descriptive Multidimensionnelle (pour les nuls).

NCI60 *Microarray gene expression profiles of the NCI 60 cell lines from 4 different platforms*

Description

The 'NCI60_4arrays' dataset from the 'omicade4' package is provided here in the MultiAssayExperiment and MultiDataSet formats. The description of the dataset provided by the omicade4 package is as follows: 'The 60 human tumour cell lines are derived from patients with leukaemia, melanoma, lung, colon, central nervous system, ovarian, renal, breast and prostate cancers. The cell line panel is widely used in anti-cancer drug screen. In this dataset, a subset of microarray gene expression of the NCI 60 cell lines from four different platforms are provided.'

Format

A list with the following elements:

- mae dataset in the MultiAssayExperiment format
- mds dataset in the MultiDataSet format

Value

List containing the NCI60_4arrays dataset from the omicade4 package in the MultiAssayExperiment and MultiDataSet formats.

References

Reinhold WC, Sunshine M, Liu H, Varma S, Kohn KW, Morris J, Doroshow J, Pommier Y (2012). CellMiner: A Web-Based Suite of Genomic and Pharmacologic Tools to Explore Transcript and Drug Patterns in the NCI-60 Cell Line Set. Cancer Research. DOI: 10.1158/0008-5472.CAN-12-1370

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

Description

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

Usage

```
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,
  plotSubC = NA,
  fig.pdfC = c("none", "interactive", "myfile.pdf")[2],
  info.txtC = c("none", "interactive", "myfile.txt")[2]
)

## 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,
  plotSubC = NA,
  fig.pdfC = c("none", "interactive", "myfile.pdf")[2],
  info.txtC = c("none", "interactive", "myfile.txt")[2]
)

## S4 method for signature 'data.frame'
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,
  plotSubC = NA,
  fig.pdfC = c("none", "interactive", "myfile.pdf")[2],
```

```
  info.txtC = c("none", "interactive", "myfile.txt")[2]
)

## S4 method for signature 'SummarizedExperiment'
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,
  plotSubC = NA,
  fig.pdfC = c("none", "interactive", "myfile.pdf")[2],
  info.txtC = c("none", "interactive", "myfile.txt")[2]
)

## S4 method for signature 'MultiAssayExperiment'
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,
  plotSubC = NA,
  fig.pdfC = c("none", "interactive", "myfile.pdf")[2],
  info.txtC = c("none", "interactive", "myfile.txt")[2]
)

## S4 method for signature 'ExpressionSet'
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,
  plotSubC = NA,
  fig.pdfC = c("none", "interactive", "myfile.pdf")[2],
  info.txtC = c("none", "interactive", "myfile.txt")[2]
)

## S4 method for signature 'MultiDataSet'
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,
  plotSubC = NA,
  fig.pdfC = c("none", "interactive", "myfile.pdf")[2],
  info.txtC = c("none", "interactive", "myfile.txt")[2]
)
```

Arguments

x	Numerical matrix, (observations x variables; NAs are allowed), data.frame, SummarizedExperiment or ExpressionSet object
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 a SummarizedExperiment or 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 componenents 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 peformed; 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)

`plotSubC` Character: Graphic subtitle

`fig.pdfC` Character: File name with '.pdf' extension for the figure; if 'interactive' (default), figures will be displayed interactively; if 'none', no figure will be generated

`info.txtC` Character: File name with '.txt' extension for the printed results (call to sink()'); if 'interactive' (default), messages will be printed on the screen; if 'none', no verbose will be generated

Value

An S4 object of class 'opls' containing the slots described below; in case x is a SummarizedExperiment, a SummarizedExperiment is returned, with the 'opls' object included in the metadata:

- 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

Eriksson et al. (2006). Multi- and Megarvariate Data Analysis. Umetrics Academy. Rosipal and Kramer (2006). Overview and recent advances in partial least squares Tenenhaus (1990). La regression PLS : theorie et pratique. Technip. Wehrens (2011). Chemometrics with R. Springer. Wold et al. (2001). PLS-regression: a basic tool of chemometrics

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[, 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)

## Application to a SummarizedExperiment

sac.se <- sacurine[["se"]]
sac.se <- opls(sac.se, "gender")
SummarizedExperiment::colData(sac.se)
SummarizedExperiment::rowData(sac.se)
sac_gender.plsda <- sac.se@metadata[["opls"]][["gender_PLSDA"]]
plot(sac_gender.plsda, typeVc = "x-score")

## Application to a MultiAssayExperiment
```

```
data(NCI60)
nci.mae <- NCI60[["mae"]]
# Restricting to the 'ME' and 'LE' cancer types and to the 'agilent' and 'hgu95' datasets
library(MultiAssayExperiment)
nci.mae <- nci.mae[, nci.mae$cancer %in% c("ME", "LE"), c("agilent", "hgu95")]
# Principal Component Analysis of each data set
nci.mae <- opls(nci.mae)
# Coloring the score plots according to cancer types
for (set.c in names(nci.mae))
plot(getOpls(nci.mae)[[set.c]][["PCA"]],
parAsColFcVn = MultiAssayExperiment::colData(nci.mae)[, "cancer"],
typeVc = "x-score",
plotSubC = set.c)
# Building PLS-DA models for the cancer type, and getting back the updated MultiDataSet
nci.mae <- opls(nci.mae, "cancer", predI = 2)
# Viewing the new variable metadata (including VIP and coefficients)
lapply(names(nci.mae), function(set.c) head(SummarizedExperiment::rowData(nci.mae[[set.c]])))

## Application to an ExpressionSet

sacSet <- sacurine[["eset"]]

sacPlsda <- opls(sacSet, "gender")
sacSet <- getEset(sacPlsda)
head(Biobase::pData(sacSet))
head(Biobase::fData(sacSet))

## Application to a MultiDataSet

data(NCI60)
nci.mds <- NCI60[["mds"]]
# Restricting to the 'agilent' and 'hgu95' datasets
nci.mds <- nci.mds[, c("agilent", "hgu95")]
# Restricting to the 'ME' and 'LE' cancer types
sampleNamesVc <- Biobase::sampleNames(nci.mds[["agilent"]])
cancerTypeVc <- Biobase::pData(nci.mds[["agilent"]])[, "cancer"]
nci.mds <- nci.mds[sampleNamesVc[cancerTypeVc %in% c("ME", "LE")], ]
# Principal Component Analysis of each data set
nci.pca <- opls(nci.mds)
# Coloring the Score plot according to cancer types
plot(nci.pca, parAsColFcVn = Biobase::pData(nci.mds[["agilent"]])[, "cancer"], typeVc = "x-score")
# Getting the updated MultiDataSet (now including scores and loadings)
nci.mds <- getMset(nci.pca)
# Building PLS-DA models for the cancer type, and getting back the updated MultiDataSet
nci.plsda <- opls(nci.mds, "cancer", predI = 2)
nci.mds <- getMset(nci.plsda)
# Viewing the new variable metadata (including VIP and coefficients)
lapply(Biobase::fData(nci.mds), head)
```

---

opls-class *Class "opls"*

Description

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, 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)

`coefficientMN` (O)PLS(-DA): Numerical matrix of regression coefficients (B; dimensions: ncol(x) x number of responses; B = W*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: 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 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 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

eset ExpressionSet: when 'opls' has been applied to an ExpressionSet, the instance with additional columns in pData containing the scores and predictions, and in fData containing the loadings, VIP, coefficients is stored here

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[, 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)
```

`oplsMultiDataSet-class` *Class "oplsMultiDataSet"*

Description

An S4 class to store PCA and (O)PLS(-DA) models generated by the application of `opls` to a MultiDataSet

Slots

`oplsLs` List: List of instances from the 'opls' class corresponding to the models built on each ExpresssionSet

Objects from the Class

Objects can be created by calls of the form `new("oplsMultiDataSet")` or by applying the `opls` function to a MultiDataSet instance

Author(s)

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

See Also

`opls`

Examples

```
# In progress
```

plot,oplsMultiDataSet,ANY-method
Plot Method for (O)PLS(-DA)

Description

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

Usage

```
## S4 method for signature 'oplsMultiDataSet,ANY'
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),
  parEllipsesL = NA,
  parLabVc = NA,
  parPaletteVc = NA,
  parTitleL = TRUE,
  parCexMetricN = NA,
  plotSubC = "",
  fig.pdfC = c("none", "interactive", "myfile.pdf")[2],
  info.txtC = c("none", "interactive", "myfile.txt")[2]
)

## S4 method for signature 'opls,ANY'
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),
  parEllipsesL = NA,
  parLabVc = NA,
  parPaletteVc = NA,
  parTitleL = TRUE,
  parCexMetricN = NA,
  plotSubC = "",
  fig.pdfC = c("none", "interactive", "myfile.pdf")[2],
```

```
  info.txtC = c("none", "interactive", "myfile.txt")[2]
)
```

Arguments

x	An S4 object of class opls or oplsMultiDataSet, 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 corresp. 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)
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]
parPaletteVc	Optional character vector of colors to be used in the plots
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
parCexMetricN	Numeric: magnification of the metrics at the bottom of score plot (default -NA- is 1 in 1x1 and 0.7 in 2x2 display)
plotSubC	Character: Graphic subtitle
fig.pdfC	Character: File name with '.pdf' extension for the figure; if 'interactive' (default), figures will be displayed interactively; if 'none', no figure will be generated
info.txtC	Character: File name with '.txt' extension for the printed results (call to sink()'); if 'interactive' (default), messages will be printed on the screen; if 'none', no verbose will be generated

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

    plsModel <- 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,
                     info.txtC = "none",
                     fig.pdfC = "none")

    plot(plsModel, typeVc = typeC)

}

sacPlsda <- opls(dataMatrix, sampleMetadata[, "gender"])
plot(sacPlsda, parPaletteVc = c("green4", "magenta"))

detach(sacurine)

#### Application to an opls object generated by an ExpressionSet

sacSet <- sacurine[["eset"]]

sacPlsda <- opls(sacSet, "gender")
plot(sacPlsda, typeVc = "x-score")

#### Application to a opls object generated by an MultiDataSet

data(NCI60)
nciMset <- NCI60[["mds"]]
# Restricting to the 'agilent' and 'hgu95' datasets
nciMset <- nciMset[, c("agilent", "hgu95")]
# Restricting to the 'ME' and 'LE' cancer types
sampleNamesVc <- Biobase::sampleNames(nciMset[["agilent"]])
cancerTypeVc <- Biobase::pData(nciMset[["agilent"]])[, "cancer"]
nciMset <- nciMset[sampleNamesVc[cancerTypeVc %in% c("ME", "LE")], ]
# Building PLS-DA models for the cancer type
nciPlsda <- opls(nciMset, "cancer", predI = 2)
```

```
plot(nciPlsda, typeVc = "x-score")
```

---

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

Description

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

Usage

```
## 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'.

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)

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)
```

`print,opls-method` *Print method for 'opls' objects*

Description

Displays information about the dataset and the model.

Usage

```
## S4 method for signature 'opls'
print(x)
```

Arguments

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

Value

Invisible.

Examples

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

print(sacurine.plsda)

detach(sacurine)
```

`residuals` *Residuals method for (O)PLS models*

Description

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

Usage

```
residuals(object, ...)
```

Arguments

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

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)
- se dataset in the SummarizedExperiment format
- eset dataset in the ExpressionSet format

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 variable 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

Thevenot E.A., Roux A., Xu Y., Ezan E. and Junot C. (2015). Analysis of the human adult urinary metabolome variations with age, body mass index and gender by implementing a comprehensive 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)
```

`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

```
tested(object)

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

Arguments

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

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,
                       responseFc,
                       subset = "odd")
```

```
trainVi <- getSubsetVi(sacurine.plsda)

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

detach(sacurine)
```

---

toW4M *Exporting a SummarizedExperiment or 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(x, filePrefixC = paste0(getwd(), "/out_"), verboseL = TRUE)

## S4 method for signature 'ExpressionSet'
toW4M(x, filePrefixC = paste0(getwd(), "/out_"), verboseL = TRUE)
```

Arguments

`x`	An S4 object of class `SummarizedExperiment` or `ExpressionSet` function.
`filePrefixC`	Character: common prefix (including repository full path) of the three file names: for example, the 'c:/mydata/setname' value will result in writting 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

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

view *view*

Description

Numeric and graphical display of a matrix, a dataframe, an ExpressionSet or a SummarizedExperiment

Display of the class, mode, size and first...last values from the object; used inside the 'view' wrapper method

Wrapper of the stats::image function used inside the 'view' method

Usage

```
view(
  x,
  printL = TRUE,
  plotL = TRUE,
  mainC = "",
  subC = "",
  paletteC = c("heat", "revHeat", "grey", "revGrey", "palette", "ramp")[1],
  rowAllL = FALSE,
  rowCexN = 1,
  rowMarN = 5.1,
  rowLabC = "",
  rowTruncI = 0,
  colAllL = FALSE,
  colCexN = 1,
  colMarN = 3.1,
  colLabC = "",
  colTruncI = 0,
  drawScaleL = TRUE,
  delimitReplicatesL = FALSE,
  standardizeL = FALSE,
  fig.pdfC = "interactive"
)

## S4 method for signature 'SummarizedExperiment'
view(
  x,
  printL = TRUE,
  plotL = TRUE,
  mainC = "",
  paletteC = c("heat", "revHeat", "grey", "revGrey", "palette", "ramp")[1],
  rowAllL = FALSE,
  rowCexN = 1,
  rowMarN = 5.1,
```

```
  rowLabC = "",
  rowTruncI = 0,
  colAllL = FALSE,
  colCexN = 1,
  colMarN = 3.1,
  colLabC = "",
  colTruncI = 0,
  drawScaleL = TRUE,
  delimitReplicatesL = FALSE,
  standardizeL = FALSE,
  fig.pdfC = "interactive"
)

## S4 method for signature 'ExpressionSet'
view(
  x,
  printL = TRUE,
  plotL = TRUE,
  mainC = "",
  paletteC = c("heat", "revHeat", "grey", "revGrey", "palette", "ramp")[1],
  rowAllL = FALSE,
  rowCexN = 1,
  rowMarN = 5.1,
  rowLabC = "",
  rowTruncI = 0,
  colAllL = FALSE,
  colCexN = 1,
  colMarN = 3.1,
  colLabC = "",
  colTruncI = 0,
  drawScaleL = TRUE,
  delimitReplicatesL = FALSE,
  standardizeL = FALSE,
  fig.pdfC = "interactive"
)

## S4 method for signature 'data.frame'
view(
  x,
  printL = TRUE,
  plotL = TRUE,
  mainC = "",
  subC = "",
  paletteC = c("heat", "revHeat", "grey", "revGrey", "palette", "ramp")[1],
  rowAllL = FALSE,
  rowCexN = 1,
  rowMarN = 5.1,
  rowLabC = "",
```

```
  rowTruncI = 0,
  colAllL = FALSE,
  colCexN = 1,
  colMarN = 3.1,
  colLabC = "",
  colTruncI = 0,
  drawScaleL = TRUE,
  delimitReplicatesL = FALSE,
  standardizeL = FALSE,
  fig.pdfC = "interactive"
)

## S4 method for signature 'matrix'
view(
  x,
  printL = TRUE,
  plotL = TRUE,
  mainC = "",
  subC = "",
  paletteC = c("heat", "revHeat", "grey", "revGrey", "palette", "ramp")[1],
  rowAllL = FALSE,
  rowCexN = 1,
  rowMarN = 5.1,
  rowLabC = "",
  rowTruncI = 0,
  colAllL = FALSE,
  colCexN = 1,
  colMarN = 3.1,
  colLabC = "",
  colTruncI = 0,
  drawScaleL = TRUE,
  delimitReplicatesL = FALSE,
  standardizeL = FALSE,
  fig.pdfC = "interactive"
)

strF(tableMF, borderI = 2, bigMarkC = ",")

imageF(
  x,
  mainC = "",
  subC = "",
  paletteC = c("heat", "revHeat", "grey", "revGrey", "palette", "ramp")[1],
  rowAllL = FALSE,
  rowCexN = 1,
  rowMarN = 5.1,
  rowLabC = "",
  rowTruncI = 0,
```

```
  colAllL = FALSE,
  colCexN = 1,
  colMarN = 1.1,
  colLabC = "",
  colTruncI = 0,
  drawScaleL = TRUE,
  delimitReplicatesL = FALSE,
  standardizeL = FALSE,
  fig.pdfC = "interactive"
)
```

Arguments

x	object to be viewed
printL	should the numerical summary be printed?
plotL	should the graphical image be displayed?
mainC	character: plot main title
subC	character(1): plot subtitle
paletteC	character: color palette; either 'heat' [default], 'revHeat', 'grey', 'revGrey', 'palette', 'ramp'
rowAllL	logical: should all rownames be displayed or only the first and last ones?
rowCexN	numeric: size of row labels [default: 1]
rowMarN	numeric: row margin [default: 5.1]
rowLabC	character: label for the y (row) axis
rowTruncI	integer: number of character for truncation of rownames (default, 0, means no truncation)
colAllL	logical: should all column names be displayed or only the first and last ones?
colCexN	numeric: size of column labels [default: 1]
colMarN	numeric: column margin [default: 3.1]
colLabC	character: label for the x (column) axis
colTruncI	integer: number of character for truncation of colnames (default, 0, means no truncation)
drawScaleL	logical: should the color scale be drawn? [default: TRUE]
delimitReplicatesL	logical: should lines be added to the image to delimit replicates in row or column names?
standardizeL	Logical: should columns be standardized for display? (i.e. subtracting the mean and dividing by the standard deviation) [default: FALSE]
fig.pdfC	character: either 'interactive' [default] or the name of the pdf file to save the figure
tableMF	Input matrix, dataframe or vector
borderI	Number of border (first and last) rows and columns to display
bigMarkC	Big mark separator for summary results

Value

this method has no output

This function has no output.

See Also

str, view

image, view

Examples

```
library(ropls)

# Get the sacurine dataset

data(sacurine)

# Display the data matrix

view(sacurine[["dataMatrix"]])
view(sacurine[["dataMatrix"]][, 1:40], mainC = "'Sacurine' dataset", rowAllL = TRUE,
colAllL = TRUE, colTruncI = 13, colMarN = 7)
view(sacurine[["dataMatrix"]][, 1:40], mainC = "'Sacurine' dataset", paletteC = "ramp")

# Display the sample metadata (dataframe)

view(sacurine[["sampleMetadata"]])

# Display the SummarizedExperiment

view(sacurine[["se"]])

# Display the ExpressionSet

view(sacurine[["eset"]])


data(sacurine)
strF(sacurine[['dataMatrix']])
strF(sacurine[['sampleMetadata']])


data(sacurine)
imageF(sacurine[['dataMatrix']])
```

Index