Package 'biosigner'

April 15, 2020

Type Package

Title Signature discovery from omics data

Version 1.14.4

Date 2020-03-18

Author Philippe Rinaudo <phd.rinaudo@gmail.com>, Etienne Thevenot <etienne.thevenot@cea.fr>

Maintainer Philippe Rinaudo <phd.rinaudo@gmail.com>, Etienne Thevenot <etienne.thevenot@cea.fr>

biocViews Classification, FeatureExtraction, Transcriptomics, Proteomics, Metabolomics, Lipidomics

Description Feature selection is critical in omics data analysis to extract restricted and meaningful molecular signatures from complex and high-dimension data, and to build robust classifiers. This package implements a new method to assess the relevance of the variables for the prediction performances of the classifier. The approach can be run in parallel with the PLS-DA, Random Forest, and SVM binary classifiers. The signatures and the corresponding 'restricted' models are returned, enabling future predictions on new datasets. A Galaxy implementation of the package is available within the Workflow4metabolomics.org online infrastructure for computational metabolomics.

Depends Biobase, ropls

Imports methods, e1071, MultiDataSet, randomForest

Suggests BioMark, BiocGenerics, BiocStyle, golubEsets, hu6800.db, knitr, omicade4, rmarkdown, testthat

VignetteBuilder knitr

License CeCILL

LazyLoad yes

NeedsCompilation no

RoxygenNote 7.0.2

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

git_branch RELEASE_3_10

git_last_commit 3776fad

git_last_commit_date 2020-03-18

Date/Publication 2020-04-14

R topics documented:

`biosigner-package` *Molecular signature discovery from omics data*

Description

Feature selection is critical in omics data analysis to extract restricted and meaningful molecular signatures from complex and high-dimension data, and to build robust classifiers. This package implements a new method to assess the relevance of the variables for the prediction performances of the classifier. The approach can be run in parallel with the PLS-DA, Random Forest, and SVM binary classifiers. The signatures and the corresponding 'restricted' models are returned, enabling future predictions on new datasets. A Galaxy implementation of the package is available within the Workflow4metabolomics.org online infrastructure for computational metabolomics.

Author(s)

Philippe Rinaudo <phd.rinaudo@gmail.com> and Etienne Thevenot <etienne.thevenot@cea.fr>.

Maintainer: Philippe Rinaudo <phd.rinaudo@gmail.com>

Examples

```
## loading the diaplasma dataset

data(diaplasma)
attach(diaplasma)

## restricting to a smaller dataset for this example

featureSelVl <- variableMetadata[, "mzmed"] >= 490 & variableMetadata[, "mzmed"] < 500
dataMatrix <- dataMatrix[, featureSelVl]
variableMetadata <- variableMetadata[featureSelVl, ]

## signature selection for all 3 classifiers
## a bootI = 5 number of bootstraps is used for this example
## we recommend to keep the default bootI = 50 value for your analyzes
```

```
set.seed(123)
diaSign <- biosign(dataMatrix, sampleMetadata[, "type"], bootI = 5)

detach(diaplasma)
```

---

biosign,MultiDataSet-method

Builds the molecular signature.

Description

Main function of the 'biosigner' package. For each of the available classifiers (PLS-DA, Random Forest, and SVM), the significant features are selected and the corresponding models are built.

Usage

```
## S4 method for signature 'MultiDataSet'
biosign(
  x,
  y,
  seedI = NULL,
  fig.pdfC = c("none", "interactive", "myfile.pdf")[2],
  info.txtC = c("none", "interactive", "myfile.txt")[2],
  ...
)

## S4 method for signature 'ExpressionSet'
biosign(x, y, ...)

## S4 method for signature 'data.frame'
biosign(x, y, ...)

## S4 method for signature 'matrix'
biosign(
  x,
  y,
  methodVc = c("all", "plsda", "randomforest", "svm")[1],
  bootI = 50,
  pvalN = 0.05,
  permI = 1,
  fixRankL = FALSE,
  seedI = 123,
  plotSubC = NA,
  fig.pdfC = c("none", "interactive", "myfile.pdf")[2],
  info.txtC = c("none", "interactive", "myfile.txt")[2],
  printL = TRUE,
  plotL = TRUE,
  .sinkC = NULL,
  ...
)
```

Arguments

x	Numerical data frame or matrix (observations x variables), or ExpressionSet object with non empty assayData and phenoData; NAs are allowed for PLS-DA but for SVM, samples with NA will be removed
y	Two-level factor corresponding to the class labels, or a character indicating the name of the column of the pData to be used, when x is an ExpressionSet object
seedI	integer: optional seed to obtain exactly the same signature when rerunning biosigner; default is '123'; set to NULL to prevent seed setting
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
...	Currently not used.
methodVc	Character vector: Either one or all of the following classifiers: Partial Least Squares Discriminant Analysis ('plsda'), or Random Forest ('randomforest'), or Support Vector Machine ('svm')
bootI	Integer: Number of bootstaps for resampling
pvalN	Numeric: To speed up the selection, only variables which significantly improve the model up to two times this threshold (to take into account potential fluctuations) are computed
permI	Integer: Random permutation are used to assess the significance of each new variable included into the model (forward selection)
fixRankL	Logical: Should the initial ranking be computed with the full model only, or as the median of the ranks from the models built on the sampled dataset?
plotSubC	Character: Graphic subtitle
printL	Logical: deprecated: use the 'info.txtC' argument instead
plotL	Logical: deprecated: use the 'fig.pdfC' argument instead
.sinkC	Character: deprecated: use the 'info.txtC' argument instead

Value

An S4 object of class 'biosign' containing the following slots: 1) 'methodVc' character vector: selected classifier(s) ('plsda', 'randomforest', and/or 'svm'), 2) 'accuracyMN' numeric matrix: balanced accuracies for the full models, and the models restricted to the 'S' and 'AS' signatures (predictions are obtained by using the resampling scheme selected with the 'bootI' and 'crossvalI' arguments), 3) 'tierMC' character matrix: contains the tier ('S', 'A', 'B', 'C', 'D', or 'E') of each feature for each classifier (features with tier 'S' have been found significant in all backward selections; features with tier 'A' have been found significant in all but the last selection, and so on), 4) modelLs list: selected classifier(s) trained on the subset restricted to the 'S' features, 5) signatureLs list: 'S' signatures for each classifier; and 6) 'AS' list: 'AS' signatures and corresponding trained classifiers, in addition to the dataset restricted to tiers 'S' and 'A' ('xMN') and the labels ('yFc')

Author(s)

Philippe Rinaudo and Etienne Thevenot (CEA)

See Also

predict.biosign, plot.biosign

Examples

```
## loading the diaplasma dataset

data(diaplasma)
attach(diaplasma)

## restricting to a smaller dataset for this example

featureSelVl <- variableMetadata[, "mzmed"] >= 490 & variableMetadata[, "mzmed"] < 500
dataMatrix <- dataMatrix[, featureSelVl]
variableMetadata <- variableMetadata[featureSelVl, ]

# signature selection for all 3 classifiers
# a bootI = 5 number of bootstraps is used for this example
# we recommend to keep the default bootI = 50 value for your analyzes

diaSign <- biosign(dataMatrix, sampleMetadata[, "type"], bootI = 5)

## Application to an ExpressionSet

diaSet <- ExpressionSet(assayData = t(dataMatrix),
                        phenoData = new("AnnotatedDataFrame",
                                        data = sampleMetadata),
                        featureData = new("AnnotatedDataFrame",
                                          data = variableMetadata),
                        experimentData = new("MIAME",
                                             title = "diaplasma"))

diaSign <- biosign(diaSet, "type", bootI = 5)
diaSet <- getEset(diaSign)
head(fData(diaSet))

detach(diaplasma)

## Application to a MultiDataSet

# Loading the 'NCI60_4arrays' from the 'omicade4' package
data("NCI60_4arrays", package = "omicade4")
# Selecting two of the four datasets
setNamesVc <- c("agilent", "hgu95")
# Creating the MultiDataSet instance
nciMset <- MultiDataSet::createMultiDataSet()
# Adding the two datasets as ExpressionSet instances
for (setC in setNamesVc) {
  # Getting the data
  exprMN <- as.matrix(NCI60_4arrays[[setC]])
  pdataDF <- data.frame(row.names = colnames(exprMN),
                        cancer = substr(colnames(exprMN), 1, 2),
                        stringsAsFactors = FALSE)
  fdataDF <- data.frame(row.names = rownames(exprMN),
                        name = rownames(exprMN),
```

```
                        stringsAsFactors = FALSE)
  # Building the ExpressionSet
  eset <- Biobase::ExpressionSet(assayData = exprMN,
                                 phenoData = new("AnnotatedDataFrame",
                                                 data = pdataDF),
                                 featureData = new("AnnotatedDataFrame",
                                                   data = fdataDF),
                                 experimentData = new("MIAME",
                                                      title = setC))
  # Adding to the MultiDataSet
  nciMset <- MultiDataSet::add_eset(nciMset, eset, dataset.type = setC,
                                    GRanges = NA, warnings = FALSE)
}
# 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")], ]
# Summary of the MultiDataSet
nciMset
# Building PLS-DA models for the cancer type, and getting back the updated MultiDataSet
nciPlsda <- ropls::opls(nciMset, "cancer", predI = 2)
nciMset <- ropls::getMset(nciPlsda)
# Selecting the significant features for PLS-DA, RF, and SVM classifiers, and getting back the updated MultiData
nciBiosign <- biosigner::biosign(nciMset, "cancer")
nciMset <- biosigner::getMset(nciBiosign)
```

`biosign-class` *Class "biosign"*

Description

The biosigner object class

Slots

`methodVc` character vector: selected classifier(s) ('plsda', 'randomforest', or 'svm')

`accuracyMN` numeric matrix: balanced accuracies for the full models, and the models restricted to the 'S' and 'AS' signatures

`tierMC` character matrix: contains the tier ('S', 'A', 'B', 'C', 'D', or 'E') of each feature for each classifier

`yFc` factor with two levels: response factor

`modelLs` list: selected classifier(s) trained on the subset restricted to the 'S' features

`signatureLs` list: 'S' signatures for each classifier

`xSubMN` matrix: dataset restricted to the 'S' tier

`AS` list: 'AS' signatures and corresponding trained classifiers, in addition to the dataset restricted to tiers 'S' and 'A' ('xMN') and the labels ('yFc')

`eset` ExpressionSet: when 'biosign' has been applied to an ExpressionSet, the instance with additional columns in fData containing the selected features is stored here

Objects from the Class

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

Author(s)

Philippe Rinaudo and Etienne Thevenot (CEA)

See Also

`biosign`

Examples

```
## loading the diaplasma dataset

data(diaplasma)
attach(diaplasma)

## restricting to a smaller dataset for this example

featureSelVl <- variableMetadata[, "mzmed"] >= 490 & variableMetadata[, "mzmed"] < 500
dataMatrix <- dataMatrix[, featureSelVl]
variableMetadata <- variableMetadata[featureSelVl, ]

## signature selection for all 3 classifiers
## a bootI = 5 number of bootstraps is used for this example
## we recommend to keep the default bootI = 50 value for your analyzes

set.seed(123)
diaSign <- biosign(dataMatrix, sampleMetadata[, "type"], bootI = 5)

detach(diaplasma)
```

---

biosignMultiDataSet-class

Class "biosignMultiDataSet"

Description

An S4 class to store the the biosign objects generated by the application of the 'biosign' method to a MultiDataSet instance

Slots

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

Objects from the Class

Objects can be created by calls of the form `new("biosignMultiDataSet",...)` or by applying the `biosign` function to a MultiDataSet instance

See Also

biosign

Examples

```
# In progress
```

---

diaplasma *Analysis of plasma from diabetic patients by LC-HRMS*

Description

Plasma samples from 69 diabetic patients were analyzed by reversed-phase liquid chromatography coupled to high-resolution mass spectrometry (Orbitrap Exactive) in the negative ionization mode. The raw data were pre-processed with XCMS and CAMERA (5,501 features), corrected for signal drift, log10 transformed, and annotated with an in-house spectral database. The patient's age, body mass index, and diabetic type are recorded. These three clinical covariates are strongly associated, most of the type II patients being older and with a higher bmi than the type I individuals.

Format

A list with the following elements:

- dataMatrix: a 69 samples x 5,501 features matrix of numeric type corresponding to the intensity profiles (values have been log10-transformed),
- sampleMetadata: a 69 x 3 data frame, with the patients' diabetic type ('type', factor), age ('age', numeric), and body mass index ('bmi', numeric),
- variableMetadata: a 5,501 x 8 data frame, with the median m/z ('mzmed', numeric) and the median retention time in seconds ('rtmed', numeric) from XCMS, the 'isotopes' (character), 'adduct' (character) and 'pcgroups' (numeric) annotations from CAMERA, and the names of the m/z and RT matching compounds from an in-house database of pure spectra from commercial metabolites ('spiDb', character).

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.

Source

'diaplasma' dataset.

References

Rinaudo P., Boudah S., Junot C. and Thevenot E.A. (2016). biosigner: a new method for the discovery of significant molecular signatures from omics data. Frontiers in Molecular Biosciences 3. doi:10.3389/fmolb.2016.00026

getAccuracyMN *Accuracies of the full model and the models restricted to the signatures*

Description

Balanced accuracies for the full models, and the models restricted to the 'S' and 'AS' signatures

Usage

```
getAccuracyMN(object, ...)

## S4 method for signature 'biosign'
getAccuracyMN(object)
```

Arguments

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

Value

A numeric matrix containing the balanced accuracies for the full models, and the models restricted to the 'S' and 'AS' signatures (predictions are obtained by using the resampling scheme selected with the 'bootI' and 'crossvalI' arguments)

Author(s)

Philippe Rinaudo and Etienne Thevenot (CEA)

Examples

```
## loading the diaplasma dataset

data(diaplasma)
attach(diaplasma)

## restricting to a smaller dataset for this example

featureSelVl <- variableMetadata[, "mzmed"] >= 490 & variableMetadata[, "mzmed"] < 500
dataMatrix <- dataMatrix[, featureSelVl]
variableMetadata <- variableMetadata[featureSelVl, ]

## signature selection for all 3 classifiers
## a bootI = 5 number of bootstraps is used for this example
## we recommend to keep the default bootI = 50 value for your analyzes

set.seed(123)
diaSign <- biosign(dataMatrix, sampleMetadata[, "type"], bootI = 5)

## individual boxplot of the selected signatures

getAccuracyMN(diaSign)
```

```
detach(diaplasma)
```

---

getEset,biosign-method *getEset method*

Description

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

Usage

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

Arguments

object	An S4 object of class biosign, created by biosign 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 computed tiers for each feature and each classifier.

Author(s)

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

Examples

```
## loading the diaplasma dataset

data(diaplasma)
attach(diaplasma)

## building the ExpresssionSet instance

diaSet <- Biobase::ExpressionSet(assayData = t(dataMatrix),
                                 phenoData = new("AnnotatedDataFrame",
                                                 data = sampleMetadata),
                                 featureData = new("AnnotatedDataFrame",
                                                   data = variableMetadata),
                                 experimentData = new("MIAME",
                                                      title = "diaplasma"))

## restricting to a smaller dataset for this example

featureSelVl <- variableMetadata[, "mzmed"] >= 490 & variableMetadata[, "mzmed"] < 500
diaSet <- diaSet[featureSelVl, ]
```

```
## signature selection for all 3 classifiers
## a bootI = 5 number of bootstraps is used for this example
## we recommend to keep the default bootI = 50 value for your analyzes

set.seed(123)
diaSign <- biosign(diaSet, "type", bootI = 5)

diaSet <- biosigner::getEset(diaSign)
head(Biobase::pData(diaSet))
head(Biobase::fData(diaSet))

detach(diaplasma)
```

---

getMset,biosignMultiDataSet-method

getMset method

Description

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

Usage

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

Arguments

`object`	An S4 object of class `biosignMultiDataSet`, created by `biosign` function applied to a MultiDataSet

Value

An S4 object of class `MultiDataSet`.

Examples

```
# Loading the 'NCI60_4arrays' from the 'omicade4' package
data("NCI60_4arrays", package = "omicade4")
# Selecting two of the four datasets
setNamesVc <- c("agilent", "hgu95")
# Creating the MultiDataSet instance
nciMset <- MultiDataSet::createMultiDataSet()
# Adding the two datasets as ExpressionSet instances
for (setC in setNamesVc) {
  # Getting the data
  exprMN <- as.matrix(NCI60_4arrays[[setC]])
  pdataDF <- data.frame(row.names = colnames(exprMN),
                        cancer = substr(colnames(exprMN), 1, 2),
                        stringsAsFactors = FALSE)
  fdataDF <- data.frame(row.names = rownames(exprMN),
                        name = rownames(exprMN),
                        stringsAsFactors = FALSE)
```

```
  # Building the ExpressionSet
  eset <- Biobase::ExpressionSet(assayData = exprMN,
                                 phenoData = new("AnnotatedDataFrame",
                                                 data = pdataDF),
                                 featureData = new("AnnotatedDataFrame",
                                                   data = fdataDF),
                                 experimentData = new("MIAME",
                                                      title = setC))
  # Adding to the MultiDataSet
  nciMset <- MultiDataSet::add_eset(nciMset, eset, dataset.type = setC,
                                    GRanges = NA, warnings = FALSE)
}
# 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")], ]
# Summary of the MultiDataSet
nciMset
# Selecting the significant features for PLS-DA, RF, and SVM classifiers, and getting back the updated MultiData
nciBiosign <- biosigner::biosign(nciMset, "cancer")
nciMset <- biosigner::getMset(nciBiosign)
# In the updated MultiDataSet, the updated featureData now contains the cancer_biosign_'classifier' columns
# indicating the selected features
lapply(fData(nciMset), head)
```

`getSignatureLs` *Signatures selected by the models*

Description

List of 'S' (or 'S' and 'A') signatures for each classifier

Usage

```
getSignatureLs(object, tierC = c("S", "AS")[1], ...)

## S4 method for signature 'biosign'
getSignatureLs(object, tierC = c("S", "AS")[1])
```

Arguments

`object`	An S4 object of class `biosign`, created by the `biosign` function.
`tierC`	Character: defines whether signatures from the 'S' tier only (default) or the ('S' and 'A') tiers should be returned
`...`	Currently not used.

Value

List of 'S' (or 'S' and 'A') signatures for each classifier

Author(s)

Philippe Rinaudo and Etienne Thevenot (CEA)

Examples

```
## loading the diaplasma dataset

data(diaplasma)
attach(diaplasma)

## restricting to a smaller dataset for this example

featureSelVl <- variableMetadata[, "mzmed"] >= 490 & variableMetadata[, "mzmed"] < 500
dataMatrix <- dataMatrix[, featureSelVl]
variableMetadata <- variableMetadata[featureSelVl, ]

## signature selection for all 3 classifiers
## a bootI = 5 number of bootstraps is used for this example
## we recommend to keep the default bootI = 50 value for your analyzes

set.seed(123)
diaSign <- biosign(dataMatrix, sampleMetadata[, "type"], bootI = 5)

## individual boxplot of the selected signatures

getSignatureLs(diaSign)

detach(diaplasma)
```

---

plot,biosignMultiDataSet,ANY-method

Plot method for biosign signatures

Description

This function plots signatures obtained by `biosign`.

Displays classifier tiers or individual boxplots from selected features

Usage

```
## S4 method for signature 'biosignMultiDataSet,ANY'
plot(
  x,
  y,
  fig.pdfC = c("none", "interactive", "myfile.pdf")[2],
  info.txtC = c("none", "interactive", "myfile.txt")[2],
  ...
)

## S4 method for signature 'biosign,ANY'
plot(
  x,
  y,
  tierMaxC = "S",
```

```
  typeC = c("tier", "boxplot")[1],
  plotSubC = NA,
  fig.pdfC = c("none", "interactive", "myfile.pdf")[2],
  info.txtC = c("none", "interactive", "myfile.txt")[2],
  file.pdfC = NULL,
  .sinkC = NULL,
  ...
)
```

Arguments

x	An S4 object of class biosign, created by the biosign function.
y	Currently not used.
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
...	Currently not used.
tierMaxC	Character: Maximum level of tiers to display: Either 'S' and 'A', (for boxplot), or also 'B', 'C', 'D', and 'E' (for tiers) by decreasing number of selections
typeC	Character: Plot type; either 'tier' [default] displaying the comparison of signatures up to the selected 'tierMaxC' or 'boxplot' showing the individual boxplots of the features selected by all the classifiers
plotSubC	Character: Graphic subtitle
file.pdfC	Character: deprecated; use the 'fig.pdfC' argument instead
.sinkC	Character: deprecated; use the 'info.txtC' argument instead

Value

A plot is created on the current graphics device.

Author(s)

Philippe Rinaudo and Etienne Thevenot (CEA)

Examples

```
# Loading the 'NCI60_4arrays' from the 'omicade4' package
data("NCI60_4arrays", package = "omicade4")
# Selecting two of the four datasets
setNamesVc <- c("agilent", "hgu95")
# Creating the MultiDataSet instance
nciMset <- MultiDataSet::createMultiDataSet()
# Adding the two datasets as ExpressionSet instances
for (setC in setNamesVc) {
  # Getting the data
  exprMN <- as.matrix(NCI60_4arrays[[setC]])
  pdataDF <- data.frame(row.names = colnames(exprMN),
                        cancer = substr(colnames(exprMN), 1, 2),
                        stringsAsFactors = FALSE)
```

```
  fdataDF <- data.frame(row.names = rownames(exprMN),
                        name = rownames(exprMN),
                        stringsAsFactors = FALSE)
  # Building the ExpressionSet
  eset <- Biobase::ExpressionSet(assayData = exprMN,
                                 phenoData = new("AnnotatedDataFrame",
                                                 data = pdataDF),
                                 featureData = new("AnnotatedDataFrame",
                                                   data = fdataDF),
                                 experimentData = new("MIAME",
                                                      title = setC))
  # Adding to the MultiDataSet
  nciMset <- MultiDataSet::add_eset(nciMset, eset, dataset.type = setC,
                                    GRanges = NA, warnings = FALSE)
}
# 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")], ]
# Summary of the MultiDataSet
nciMset
# Selecting the significant features for PLS-DA, RF, and SVM classifiers, and getting back the updated MultiData
nciBiosign <- biosigner::biosign(nciMset, "cancer")
# Plotting the selected signatures
plot(nciBiosign)

## loading the diaplasma dataset

data(diaplasma)
attach(diaplasma)

## restricting to a smaller dataset for this example

featureSelVl <- variableMetadata[, "mzmed"] >= 490 & variableMetadata[, "mzmed"] < 500
dataMatrix <- dataMatrix[, featureSelVl]
variableMetadata <- variableMetadata[featureSelVl, ]

## signature selection for all 3 classifiers
## a bootI = 5 number of bootstraps is used for this example
## we recommend to keep the default bootI = 50 value for your analyzes

set.seed(123)
diaSign <- biosign(dataMatrix, sampleMetadata[, "type"], bootI = 5)

## individual boxplot of the selected signatures

plot(diaSign, typeC = "boxplot")

detach(diaplasma)
```

---

predict,biosign-method

Predict method for 'biosign' signature objects

Description

This function predicts values based upon biosign classifiers trained on the 'S' signature

Usage

```
## S4 method for signature 'biosign'
predict(object, newdata, tierMaxC = "S", ...)
```

Arguments

object	An S4 object of class biosign, created by biosign function.
newdata	Either a data frame or a matrix, containing numeric columns only, with column names identical to the 'x' used for model training with 'biosign'.
tierMaxC	Character: Maximum level of tiers to display: Either 'S'or 'A'.
...	Currently not used.

Value

Data frame with the predictions for each classifier as factor columns.

Author(s)

Philippe Rinaudo and Etienne Thevenot (CEA)

Examples

```
## loading the diaplasma dataset

data(diaplasma)
attach(diaplasma)

## restricting to a smaller dataset for this example

featureSelVl <- variableMetadata[, "mzmed"] >= 490 & variableMetadata[, "mzmed"] < 500
dataMatrix <- dataMatrix[, featureSelVl]
variableMetadata <- variableMetadata[featureSelVl, ]

## training the classifiers
## a bootI = 5 number of bootstraps is used for this example
## we recommend to keep the default bootI = 50 value for your analyzes

set.seed(123)
diaSign <- biosign(dataMatrix, sampleMetadata[, "type"], bootI = 5)

## fitted values (for the subsets restricted to the 'S' signatures)
sFitDF <- predict(diaSign)

## confusion tables
print(lapply(sFitDF, function(predFc) table(actual = sampleMetadata[,
"type"], predicted = predFc)))

## balanced accuracies
sapply(sFitDF, function(predFc) { conf <- table(sampleMetadata[,
"type"], predFc)
```

```
conf <- sweep(conf, 1, rowSums(conf), "/")
mean(diag(conf))
})
## note that these values are slightly different from the accuracies
## returned by biosign because the latter are computed by using the
## resampling scheme selected by the bootI or crossvalI arguments
getAccuracyMN(diaSign)["S", ]

detach(diaplasma)
```

---

show,biosign-method *Show method for 'biosign' signature objects*

Description

Prints the selected features and the accuracies of the classifiers.

Usage

```
## S4 method for signature 'biosign'
show(object)
```

Arguments

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

Value

Invisible.

Author(s)

Philippe Rinaudo and Etienne Thevenot (CEA)

Examples

```
## loading the diaplasma dataset

data(diaplasma)
attach(diaplasma)

## restricting to a smaller dataset for this example

featureSelVl <- variableMetadata[, "mzmed"] >= 490 & variableMetadata[, "mzmed"] < 500
dataMatrix <- dataMatrix[, featureSelVl]
variableMetadata <- variableMetadata[featureSelVl, ]

## signature selection for all 3 classifiers
## a bootI = 5 number of bootstraps is used for this example
## we recommend to keep the default bootI = 50 value for your analyzes

set.seed(123)
```

```
diaSign <- biosign(dataMatrix, sampleMetadata[, "type"], bootI = 5)

diaSign

detach(diaplasma)
```

Index