Package 'MSclassifR'

August 13, 2022

Type Package

Title Automated Classification of Mass Spectra

Version 0.3.0

Maintainer Alexandre Godmer <alexandre.godmer@aphp.fr>

Description
Functions to classify mass spectra in known categories, and to determine discriminant mass-over-charge values. It includes easy-to-use functions for pre-processing mass spectra, functions to determine discriminant mass-over-charge values (m/z) from a library of mass spectra corresponding to different categories, and functions to predict the category (species, phenotypes, etc.) associated to a mass spectrum from a list of selected mass-over-charge values. Three vignettes illustrating how to use the functions of this package from real data sets are also available online to help users: <https://agodmer.github.io/MSclassifR_examples/Vignettes/Vignettemsclassifr_Ecrobiav3.html>, <https://agodmer.github.io/MSclassifR_examples/Vignettes/Vignettemsclassifr_Klebsiellav3.html> and <https://agodmer.github.io/MSclassifR_examples/Vignettes/Vignettemsclassifr_DAv3.html>.

License GPL (>= 3)

Encoding UTF-8

LazyData true

Depends R (>= 4.0), cp4p, caret, statmod,

Imports e1071, MALDIquant, MALDIrppa, MALDIquantForeign, mixOmics,
reshape2, ggplot2, nnet, dplyr, fuzzyjoin, VSURF, metap,
xgboost, glmnet, performanceEstimation, mltools, mclust, UBL,
stats, limma, car,

Suggests knitr, rmarkdown,

NeedsCompilation yes

RoxygenNote 7.1.1

Author Alexandre Godmer [aut, cre],
Quentin Giai Gianetto [aut]

Repository CRAN

Date/Publication 2022-08-13 14:10:02 UTC

R topics documented:

`CitrobacterRKImetadata`

Metadata of mass spectra corresponding to the bacterial species Citrobacter *sp. from The Robert Koch-Institute (RKI) database of microbial MALDI-TOF mass spectra*

Description

Metadada of the `CitrobacterRKIspectra` list of mass spectra.

Usage

```
data("CitrobacterRKImetadata", package = "MSclassifR")
```

Format

A data frame with 14 rows (each corresponding to a mass spectrum), and five columns that contain (in order): the strain name, the species name, the spot, a sample number and the name of the strain associated with the spot.

Details

The Robert Koch-Institute (RKI) database of microbial MALDI-TOF mass spectra contains raw mass spectra. Only mass spectra of the *Citrobacter* bacterial species were collected. Metadata were manually reported from raw data.

Source

The raw data were downloaded from this link : `https://zenodo.org/record/163517#.YIkWiNZuJCp`. The dataset focuses only on mass spectra from *Citrobacter*.

References

Lasch, Peter, Stammler, Maren, & Schneider, Andy. (2018). Version 3 (20181130) of the MALDI-TOF Mass Spectrometry Database for Identification and Classification of Highly Pathogenic Microorganisms from the Robert Koch-Institute (RKI) [Data set]. Zenodo.doi: 10.5281/zenodo.163517

`CitrobacterRKIspectra` *Mass spectra corresponding to the bacterial species* Citrobacter *sp. from The Robert Koch-Institute (RKI) database of microbial MALDI-TOF mass spectra*

Description

Mass spectra of the `CitrobacterRKIspectra` dataset.

Usage

```
data("CitrobacterRKIspectra", package = "MSclassifR")

#####
#Plotting the first mass spectrum
#library("MSclassifR")
#PlotSpectra(SpectralData=CitrobacterRKIspectra[[1]],absx = "ALL", Peaks = NULL,
#            Peaks2 = NULL, col_spec = 1, col_peak = 2, shape_peak = 3,
#            col_peak2 = 2, shape_peak2 = 2)
```

Format

A list that contains 14 objects of class S4 corresponding each to a each mass spectrum.

Details

The Robert Koch-Institute (RKI) database of microbial MALDI-TOF mass spectra contains raw mass spectra. Only mass spectra of the *Citrobacter* bacterial species were collected.

Source

The raw data were downloaded from this link : `https://zenodo.org/record/163517#.YIkWiNZuJCp`. The dataset focuses only on mass spectra from *Citrobacter*.

References

Lasch, Peter, Stammler, Maren, & Schneider, Andy. (2018). Version 3 (20181130) of the MALDI-TOF Mass Spectrometry Database for Identification and Classification of Highly Pathogenic Microorganisms from the Robert Koch-Institute (RKI) [Data set]. Zenodo.doi: 10.5281/zenodo.163517

LogReg — *Estimation of a multinomial logistic regression to predict the category to which a mass spectrum belongs*

Description

This function estimates a multinomial logistic regression using cross-validation to predict the category (species, phenotypes...) to which a mass spectrum belongs from a set of shortlisted mass-over-charge values corresponding to discriminant peaks. Two main kinds of models can be estimated: linear or nonlinear (with neural networks, random forests, support vector machines with linear kernel, or eXtreme Gradient Boosting). Hyperparameters are randomly searched, except for the eXtreme Gradient Boosting where a grid search is performed.

Usage

```
LogReg(X,
       moz,
       Y,
       number = 2,
       repeats = 2,
     Metric = c("Kappa", "Accuracy", "F1", "AdjRankIndex", "MatthewsCorrelation"),
       kind="linear",
       Sampling = c(NULL, "up", "down", "smote"))
```

Arguments

`X`	`matrix` corresponding to a library of mass spectra. Each row of `X` is the intensities of a mass spectrum measured on the `moz` values. The columns should be represented by mass-over-charge values.
`moz`	`vector` with shortlisted mass-over-charge values.
`Y`	`factor` with a length equal to the number of rows in `X` and containing the categories of each mass spectrum in `X`.
`number`	`integer` corresponding to the number of folds or number of resampling iterations. See arguments of the `trainControl` function of the `caret` R package.
`Metric`	a `charater` indicating metric to select the optimal model. The possibles metrics are the `"Kappa"` coefficient,`"Accruracy"`, the `"F1"` score, `"AdjRankIndex"` for the Adjusted Rand Index or `"MatthewsCorrelation"` for the Matthews Correlation Coefficient.
`repeats`	`integer` corresponding to the number of complete sets of folds to compute. See `trainControl` function of the `caret` R package for more details.
`kind`	If `kind="nnet"`, then a nonlinear multinomial logistic regression is estimated via neural networks. If `kind="rf"`, then it is estimated via random forests. If `kind="svm"`, then it is estimated via support vector machines with linear kernel. If `kind="xgb"`, then it is estimated via eXtreme gradient boosting. Else a linear multinomial logistic regression is performed (by default).

`Sampling`	a `charater` indicating an optional subsampling method to handle imbalanced datasets. The possibles methods to rebalance the data are `"up"`, `"down"` and `"smote"`.

Details

This function estimates a model from a library of mass spectra for which we already know the category to which they belong (ex.: species, etc). This model can next be used to predict the category of a new coming spectrum for which the category is unknown (see `PredictLogReg`).

The estimation is performed using the `train` function of the `caret` R package. For each kind of model, random parameters are tested to find a model according to the best `metric`. The formulas for the `metric` are as follows:

$$Accuracy = Number of correct predictions / Total number of predictions$$

$$Kappa coefficient = Observed agreement - chance agreement / 1 - chance agreement$$

$$F1 = TruePositive / (TruePositive + 1/2(FalsePositive + FalseNegative))$$

The adjusted Rand index (`"AdjRankIndex"`) is defined as the corrected-for-chance version of the Rand index which allows you to compare two groups, see `mclust` package and `adjustedRandIndex()` function for more details. The Matthews correlation coefficient (`"MatthewsCorrelation"`) which measures the association of two variables is calculated using `mcc` function in the `mltools` R package.

For `Sampling` methods available for unbalanced data: `"up"` corresponds to the up-sampling method which consists of random sampling (with replacement) so that the minority class is the same size as the majority class; `"down"` corresponds to the down-sampling method randomly which consists of random sampling (without replacement) of the majority class so that their class frequencies match the minority class; `"smote"` correspond to the Synthetic Minority Over sampling Technique (SMOTE) specific algorithm for data augmentation which consist of creating new data from minority class using the K Nearest Neighbor algorithm.

Value

Returns a `list` with four items:

`train_mod`	a `list` corresponding to the output of the train function of the `caret` R package containing the multinomial regression model estimated using repeated cross-validation.
`conf_mat`	a confusion matrix containing percentages classes of predicted categories in function of actual categories, resulting from repeated cross-validation.
`stats_global`	a `data frame` containing the mean and standard deviation values of the "Accuracy"" and "Kappa" parameters computed for each cross-validation.
`boxplot`	a `ggplot` object (see `ggplot2` R package). This is a graphical representation of the `Metric` parameters of `stats_global` using boxplots.

References

Kuhn, M. (2008). Building predictive models in R using the caret package. Journal of statistical software, 28(1), 1-26.

L. Hubert and P. Arabie (1985) Comparing Partitions, Journal of the Classification, 2, pp. 193-218.

Scrucca L, Fop M, Murphy TB, Raftery AE (2016). mclust 5: clustering, classification and density estimation using Gaussian finite mixture models. The R Journal.

Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. 2002. SMOTE: synthetic minority over-sampling technique. J. Artif. Int. Res. 16, 1.

Matthews, B. W. (1975). "Comparison of the predicted and observed secondary structure of T4 phage lysozyme". Biochimica et Biophysica Acta (BBA) - Protein Structure. PMID 1180967.

Examples

```
library("MSclassifR")
library("MALDIquant")

###############################################################################
## 1. Pre-processing of mass spectra

# load mass spectra and their metadata
data("CitrobacterRKIspectra","CitrobacterRKImetadata", package = "MSclassifR")
# standard pre-processing of mass spectra
spectra <- SignalProcessing(CitrobacterRKIspectra)
# detection of peaks in pre-processed mass spectra
peaks <- PeakDetection(x = spectra, labels = CitrobacterRKImetadata$Strain_name_spot)
# matrix with intensities of peaks arranged in rows (each column is a mass-over-charge value)
IntMat <- MALDIquant::intensityMatrix(peaks)
rownames(IntMat) <- paste(CitrobacterRKImetadata$Strain_name_spot)
# remove missing values in the matrix
IntMat[is.na(IntMat)] <- 0
# normalize peaks according to the maximum intensity value for each mass spectrum
IntMat <- apply(IntMat,1,function(x) x/(max(x)))
# transpose the matrix for statistical analysis
X <- t(IntMat)
# define the known categories of mass spectra for the classification
Y <- factor(CitrobacterRKImetadata$Species)

###############################################################################
## 2. Selection of discriminant mass-over-charge values using sPLS-DA
# with 5 to 10 variables,
# up sampling method and
# trained with the Accuracy coefficient metric

a <- MSclassifR::SelectionVar(X,
                              Y,
                              MethodSelection = c("RFERF"),
                              MethodValidation = c("cv"),
                              PreProcessing = c("center","scale","nzv","corr"),
                              NumberCV = 2,
                              Metric = "Accuracy",
                              Sizes = c(2:5),
                              Sampling = "up")

sel_moz=a$sel_moz

###############################################################################
## 3. Perform LogReg from shortlisted discriminant mass-over-charge values

# linear multinomial regression
# without sampling mehod
# and trained with the Kappa coefficient metric

model_lm=MSclassifR::LogReg(X=X,
                            moz=sel_moz,
                            Y=factor(Y),
                            number=2,
                            repeats=2,
                            Metric = "Kappa")
# Estimated model:
model_lm

# nonlinear multinomial regression using neural networks
# with up-sampling method and
# trained with the Kappa coefficient metric

model_nn=MSclassifR::LogReg(X=X,
                            moz=sel_moz,
                            Y=factor(Y),
                            number=2,
                            repeats=2,
                            kind="nnet",
                            Metric = "Kappa",
                            Sampling = "up")
# Estimated model:
model_nn

# nonlinear multinomial regression using random forests
# without down-sampling method and
# trained with the Kappa coefficient metric

model_rf=MSclassifR::LogReg(X=X,
                            moz=sel_moz,
                            Y=factor(Y),
                            number=2,
                            repeats=2,
                            kind="rf",
                            Metric = "Kappa",
                            Sampling = "down")

# Estimated model:
model_rf
```

```
# nonlinear multinomial regression using xgboost
# with down-sampling method and
# trained with the Kappa coefficient metric

model_xgb=MSclassifR::LogReg(X=X,
                             moz=sel_moz,
                             Y=factor(Y),
                             number=2,
                             repeats=2,
                             kind="xgb",
                             Metric = "Kappa",
                             Sampling = "down")
# Estimated model:
model_xgb

# nonlinear multinomial regression using svm
# with down-sampling method and
# trained with the Kappa coefficient metric

model_svm=MSclassifR::LogReg(X=X,
                             moz=sel_moz,
                             Y=factor(Y),
                             number=2,
                             repeats=2,
                             kind="svm",
                             Metric = "Kappa",
                             Sampling = "down")
# Estimated model:
model_svm

##########
# Of note, step 3 can be performed several times
# to find optimal models
# because of random hyperparameter search

###############################################################################
## 4. Select best models in term of average Kappa and saving it for reuse

Kappa_model=c(model_lm$stats_global[1,2],model_nn$stats_global[1,2],
         model_rf$stats_global[1,2],model_xgb$stats_global[1,2],model_svm$stats_global[1,2])
names(Kappa_model)=c("lm","nn","rf","xgb","svm")
#Best models in term of accuracy
Kappa_model[which(Kappa_model==max(Kappa_model))]

#save best models for reuse
#models=list(model_lm$train_mod,model_nn$train_mod,model_rf$train_mod,
#model_xgb$train_mod,model_svm$train_mod)
#models_best=models[which(Kappa_model==max(Kappa_model))]
#for (i in 1:length(models_best)){
#save(models_best[[i]], file = paste0("model_best_",i,".rda",collapse="")
#}
```

```
#load a saved model
#load("model_best_1.rda")

################################################################################
## 5. Try other metrics to select the best model

# linear multinomial regression
# with up-sampling method and
# trained with the Adjusted Rank index metric

model_lm=MSclassifR::LogReg(X=X,
                            moz=sel_moz,
                            Y=factor(Y),
                            number=2,
                            repeats=3,
                            Metric = "AdjRankIndex",
                            Sampling = "up")
```

`MSclassifR` *Automated classification of mass spectra*

Description

This package provides R functions to classify mass spectra in known categories, and to determine discriminant mass-over-charge values. It was developed with the aim of identifying very similar species or phenotypes of bacteria from mass spectra obtained by Matrix Assisted Laser Desorption Ionisation - Time Of Flight Mass Spectrometry (MALDI-TOF MS). However, the different functions of this package can also be used to classify other categories associated to mass spectra; or from mass spectra obtained with other mass spectrometry techniques. It includes easy-to-use functions for pre-processing mass spectra, functions to determine discriminant mass-over-charge values (m/z) from a library of mass spectra corresponding to different categories, and functions to predict the category (species, phenotypes, etc.) associated to a mass spectrum from a list of selected mass-over-charge values. Three vignettes illustrating how to use the functions of this package from real data sets are also available online to help users: <https://agodmer.github.io/MSclassifR_examples/Vignettes/Vignettemsclassifr_E
<https://agodmer.github.io/MSclassifR_examples/Vignettes/Vignettemsclassifr_Klebsiellav3.html>
and <https://agodmer.github.io/MSclassifR_examples/Vignettes/Vignettemsclassifr_DAv3.html>.

Value

No return value. Package description.

Author(s)

Alexandre Godmer, Quentin Giai Gianetto

PeakDetection — *Detection of peaks in* `MassSpectrum` *objects.*

Description

This function performs a data analysis pipeline to pre-process mass spectra. It provides average intensities and detects peaks using functions of R packages `MALDIquant` and `MALDIrppa`.

Usage

```
PeakDetection(x,
              labels,
              averageMassSpectraMethod = "median",
              SNRdetection = 3,
              PeakDetectionMethod = "MAD",
              halfWindowSizeDetection = 11,
              AlignMethod = "strict",
              Tolerance = 0.002,
              ...)
```

Arguments

`x`	a `list` of `MassSpectrum` objects (see `MALDIquant` R package).
`labels`	a `list` of `factor` objects to do groupwise averaging.
`averageMassSpectraMethod`	a `character` indicating the method used to average mass spectra according to labels. It is fixed to `"median"` by default.This function can be replaced by another mathematical function such as `"mean"`. See `averageMassSpectra` of `MALDIquant` R package.
`PeakDetectionMethod`	a `character` indicating the noise estimation method. It uses `"MAD"` method for `list` of `MassSpectrum` objects. This noise estimation method estimation method can be remplaced `"SuperSmoother"`. See `estimateNoise-methods` of the `MALDIquant` R package for details.
`SNRdetection`	a `numeric` value indicating the signal-to-noise ratio used to detect peaks (by default = `3`). See `detectPeaks-methods` of the `MALDIquant` R package for details.
`halfWindowSizeDetection`	a `numeric` value half window size to detect peaks (by default = `11`). See `detectPeaks-methods` of the `MALDIquant` R package for details.
`AlignMethod`	a `character` indicating the method used to equalize masses for similar peaks. The `"strict"` method is used by default corresponding to a unique peak per bin from the same sample. This method can be remplaced by `"relaxed"` corresponding to multiple peaks per bin from the same sample. See `binPeaks` of the `MALDIquant` R package for more details.

`Tolerance`	a `numeric` value corresponding to the maximal deviation in peak masses to be considered as identical in ppm (by default = `0.002`). See `determineWarpingFunctions` of the `MALDIquant` R package for details.
`...`	other arguments from `MALDIquant` and `MALDIrppa` packages.

Details

The `PeakDetection` function provides an analysis pipeline for `MassSpectrum` objects including peaks detection and binning.

All the methods used for `PeakDetection` functions are selected from `MALDIquant` and `MALDIrppa` packages.

Value

Returns a list of `MassPeaks` objects (see `MALDIquant` R package) for each mass spectrum in `x`.

References

Gibb S, Strimmer K. MALDIquant: a versatile R package for the analysis of mass spectrometry data. Bioinformatics. 2012 Sep 1;28(17):2270-1. doi: 10.1093/bioinformatics/bts447. Epub 2012 Jul 12. PMID: 22796955.

Javier Palarea-Albaladejo, Kevin Mclean, Frank Wright, David G E Smith, MALDIrppa: quality control and robust analysis for mass spectrometry data, Bioinformatics, Volume 34, Issue 3, 01 February 2018, Pages 522 - 523, doi: 10.1093/bioinformatics/btx628

Examples

```
library("MALDIquant")
library("MALDIquantForeign")
library("MSclassifR")


## Load mass spectra and metadata
data("CitrobacterRKIspectra", "CitrobacterRKImetadata", package = "MSclassifR")

## Pre-processing of mass spectra
spectra <- SignalProcessing(CitrobacterRKIspectra)

## Detection of peaks in pre-processed mass spectra
peaks <- PeakDetection(x = spectra,
 labels = CitrobacterRKImetadata$Strain_name_spot,
 averageMassSpectraMethod = "median",
 SNRdetection = 3,
 halfWindowSizeDetection = 11,
 AlignFrequency = 0.20,
 AlignMethod = "strict",
 Tolerance = 0.002)

# Plot peaks on a pre-processed mass spectrum
PlotSpectra(SpectralData=spectra[[1]],Peaks=peaks[[1]],col_spec="blue",col_peak="black")
```

PlotSpectra *Plot mass spectra with detected peaks*

Description

This function performs a plot of a `AbstractMassObject` object (see the `MALDIquant` R package). It can be used to highlight peaks in a mass spectrum.

Usage

```
PlotSpectra(SpectralData, absx="ALL", Peaks=NULL, Peaks2=NULL, col_spec=1,
            col_peak=2, shape_peak=3, col_peak2=2, shape_peak2=2)
```

Arguments

`SpectralData`	`MassSpectrum` object of S4 class (see `MALDIquant` R package).
`absx`	`vector` indicating lower and upper bounds for the mass-over-charge values to plot.
`Peaks`	`MassPeaks` object (see `MALDIquant` R package). If NULL, peaks are not highlighted.
`Peaks2`	numeric `vector` of mass-over-charge values to plot on the mass spectrum.
`col_spec`	color of the mass spectrum.
`col_peak`	color of the peak points corresponding to `Peaks`.
`shape_peak`	shape of the peak points corresponding to `Peaks`.
`col_peak2`	color of the peak points corresponding to `Peaks2`.
`shape_peak2`	Shape of the peak points corresponding to `Peaks2`.

Value

A `ggplot` object (see `ggplot2` R package). Mass-over-charge values are in x-axis and intensities in y-axis.

Examples

```
library("MSclassifR")

# Load mass spectra
data("CitrobacterRKIspectra", package = "MSclassifR")
# Plot raw mass spectrum
PlotSpectra(SpectralData = CitrobacterRKIspectra[[1]])
# standard pre-processing of mass spectra
spectra <- SignalProcessing(CitrobacterRKIspectra)
# Plot pre-processed mass spectrum
```

```
PlotSpectra(SpectralData=spectra[[1]])
# detection of peaks in pre-processed mass spectra
peaks <- PeakDetection(x = spectra, labels = CitrobacterRKImetadata$Strain_name_spot)
# Plot peaks on pre-processed mass spectrum
PlotSpectra(SpectralData=spectra[[1]],Peaks=peaks[[1]],col_spec="blue",col_peak="black")
```

PredictFastClass — *Prediction of the category to which a mass spectrum belongs using linear regressions of mass spectra.*

Description

For each mass peak in a list of mass peaks, a linear regression is performed between the mass spectrum and mass spectra corresponding to a category. This is performed for each category and associated to an Akaike Information Criterium. Next, the AIC are used to determine the belonging of a mass spectrum to a category. It also provides a probability that the mass spectrum does not belong to any of the input categories.

Usage

```
PredictFastClass(peaks,
                 mod_peaks,
                 Y_mod_peaks,
                 moz="ALL",
                 tolerance = 6,
                 toleranceStep = 2,
                 normalizeFun = TRUE,
                 noMatch = 0)
```

Arguments

`peaks`	a list of `MassPeaks` objects (see `MALDIquant` R package).
`mod_peaks`	an intensity matrix corresponding to mass spectra for which the category is known. Each column is a mass-over-charge value, each row corresponds to a mass spectrum.
`Y_mod_peaks`	a `factor` with a length equal to the number of mass spectra in `mod_peaks` and containing the categories of each mass spectrum in `mod_peaks`.
`moz`	a `vector` with the set of shortlisted mass-over-charge values that corresponds to mass-over-charge values in the columns of `mod_peaks`. By default, all the mass-over-charge values in `mod_peaks` are used.
`tolerance`	a `numeric` value of accepted tolerance to match peaks to the set of shortlisted mass-over-charge values. It is fixed to 6 Da by default.
`toleranceStep`	a `numeric` value added to the `tolerance` parameter to match peaks to the set of shortlisted mass-over-charge values. It is fixed to 2 Da by default.

`normalizeFun`	a `logical` value, if `TRUE` (default) the maximum intensity will be equal to 1, the other intensities will be expressed in ratio to this maximum.
`noMatch`	a `numeric` value used to replace intensity values if there is no match detected between peaks and the set of shortlisted mass-over-charge values `moz`. It is fixed to 0 by default.

Value

Returns a `dataframe` containing AIC criteria by category for each mass spectrum in `peaks`. The AIC criterion should be minimal for the most probable category. The `pred_cat` column is the predicted category for each mass spectrum in `peaks`. The `p_not_in_DB` is the minimal p-value of several Fisher tests testing if all the linear coefficients associated to mass spectra of a category are null. It can be interpreted as a p-value that the mass spectrum is not present in the input database.

Examples

```
library("MSclassifR")
library("MALDIquant")

#Test on Ecrobia
## url for load the  MS data
MSdataE <- "https://agodmer.github.io/MSData/Ecrobia/MassSpectra_Ecrobia.Rdata"
MSMetadataE <- "https://agodmer.github.io/MSData/Ecrobia/metaData_Ecrobia.Rdata"
## Load mass spectra
load(url(MSdataE))
load(url(MSMetadataE))

## Split data to tune the model according train set
Y <- factor(metaData_Ecrobia$Species)
Index_skfold <- caret::createDataPartition(Y, p = 0.8, list=FALSE)

## Train set creation
Xtrain <- MassSpectra_Ecrobia[Index_skfold]
Y_train <- Y[Index_skfold]

spectra <- MSclassifR::SignalProcessing(Xtrain,alignSpectra_SN = 2)

## Peaks detection for the train set
peaks <- MSclassifR::PeakDetection(spectra, labels = c(1:length(Xtrain)))

## Perfom an Intensity Matrix
IntMat <- MALDIquant::intensityMatrix(peaks)
## Rows are named according to selected metadata
rownames(IntMat) <-  paste(c(1:length(Xtrain)))
# remove missing values in the matrix
IntMat[is.na(IntMat)] <- 0
# normalize peaks according to the maximum intensity value for each mass spectrum
IntMat <- apply(IntMat,1,function(x) x/(max(x)))
```

```
## Transpose Matrix for the train set
X_train <- t(IntMat)

## Test set creation
Xtest <- MassSpectra_Ecrobia[-Index_skfold]
Y_test <- Y[-Index_skfold]
spectra2 <- MSclassifR::SignalProcessing(Xtest,alignSpectra_SN = 2)

## Peaks detection for the test set
X_test <- MSclassifR::PeakDetection(spectra2, labels = c(1:length(Xtest)))

#Predict species without peak selection using a tolerance of 3 Da
res = PredictFastClass(peaks=X_test,
                       mod_peaks=X_train,
                       Y_mod_peaks=Y_train,
                       tolerance = 3)

#comparing predicted categories (species) and the truth
cbind(res$pred_cat,as.character(Y_test))

# The method can be applied after a peak selection step
a <- SelectionVar(X_train,
                  Y_train,
                  MethodSelection = c("RFERF"),
                  MethodValidation = c("cv"),
                  PreProcessing = c("center","scale","nzv","corr"),
                  NumberCV = 2,
                  Metric = "Kappa",
                  Sizes = c(20:40),
                  Sampling = "up")

#Predict species from selected peaks using a tolerance of 3 Da
res = PredictFastClass(peaks=X_test,
                       moz = a$sel_moz,
                       mod_peaks=X_train,
                       Y_mod_peaks=Y_train, tolerance = 3)

#comparing predicted categories (species) and the truth
cbind(res$pred_cat,as.character(Y_test))
```

PredictLogReg — *Prediction of the category to which a mass spectrum belongs from a multinomial logistic regression model*

Description

This function predicts the category (species, phenotypes...) to which a mass spectrum belongs from a set of shortlisted mass-over-charge values of interest and a short-listed multinomial logistic regression model (see LogReg).

Usage

```
PredictLogReg(peaks,
              model,
              moz,
              tolerance = 6,
              toleranceStep = 2,
              normalizeFun = TRUE,
              noMatch=0,
              Reference = NULL)
```

Arguments

`peaks`	a list of `MassPeaks` objects (see `MALDIquant` R package).
`model`	a model or a list of models estimated from a set of shortlisted mass-over-charge values (output of the `LogReg` function).
`moz`	a `vector` with the set of shortlisted mass-over-charge values used to estimate the model `Model`.
`tolerance`	a `numeric` value of accepted tolerance to match peaks to the set of shortlisted mass-over-charge values. It is fixed to 6 Da by default.
`toleranceStep`	a `numeric` value added to the `tolerance` parameter to match peaks to the set of shortlisted mass-over-charge values. It is fixed to 2 Da by default.
`normalizeFun`	a `logical` value, if `TRUE` (default) the maximum intensity will be equal to 1, the other intensities will be expressed in ratio to this maximum.
`noMatch`	a `numeric` value used to replace intensity values if there is no match detected between peaks and the set of shortlisted mass-over-charge values `moz`. It is fixed to 0 by default.
`Reference`	a `factor` with a length equal to the number of rows in `X` and containing the categories of each mass spectrum in `X`. `"NULL"` by default.

Details

The `PredictLogReg` function allows to predict the membership of a mass spectrum to a category from a multinomial logistic regression model. The mass spectrum from the `peaks` object will be matched to the discriminant mass-over-chage (m/z) values (`sel_moz` object from the `SelectionVar` function) with a tolerance between two m/z defined by the `tolerance` parameter (by default this value is 6 Da). If a repetition of same m/z occurs in the selection, only the m/z that is closest in mass peaks (`moz`) is used. When no match, intensity values are replaced by the `noMatch` argument. If no m/z values from `peaks` object matched with the m/z in the object `moz`, the tolerance will be increased according to a numeric value defined in the `toleranceStep` parameter and a warning will be notified. Note that it is possible to not perform the `SelectionVar` function prior to the `PredictLogReg` function, and to replace the argument `moz` by all the m/z values present in a mass spectrum.

Value

Returns a `dataframe` containing probabilities of membership by category for each mass spectrum in `peaks`. The method used is provided in the `method` column. The `comb_fisher` method is the result of the Fisher's method when merging probabilities of membership of used prediction models.The `max_vote` method is the result of the maximum voting from used prediction models.

If the `Reference` parameter is not null, the function returns:

`Confusion.Matrix`	a `list` of confusion matrix (cross-tabulation with associated statitics) corresponding to the output of the `confusionMatrix` function of the `caret` R package.
`Gobal.stat`	a `data.frame` with three columns corresponding to the value (`value` column) of a statistic parameter (`Statistic.parameter` column) from a method used (`model` column) obtained with the `LogReg` function. See `LogReg` function for the Statistic.parameter column.
`Details.stat`	a `data.frame` with four columns corresponding to the same as `Gobal.stat` dataframe with the class concerned for estimated statistic parameter (class column). All statistic parameters are extracted from the output of the `confusionMatrix` function of the `caret` R package.
`Correct.ClassificationFreq`	a `data.frame` with predicted class (Prediction column) from a method (Model column) and the reference of the categories of each mass spectrum (Reference column). The `Freq` column indicates the number of times the category was correctly predicted by the method.
`Incorrect.ClassificationFreq`	a `data.frame` with predicted class (Prediction column) from a method (`Model` column) and the reference of the categories of each mass spectrum (`Reference` column). The `Freq` column indicates the number of times the category was not correctly predicted by the method.

References

Kuhn, M. (2008). Building predictive models in R using the caret package. Journal of statistical software, 28(1), 1-26.

Examples

```
library("MSclassifR")
library("MALDIquant")

###############################################################################
## 1. Pre-processing of mass spectra

# load mass spectra and their metadata
data("CitrobacterRKIspectra","CitrobacterRKImetadata", package = "MSclassifR")
# standard pre-processing of mass spectra
spectra <- SignalProcessing(CitrobacterRKIspectra)
# detection of peaks in pre-processed mass spectra
peaks <- PeakDetection(x = spectra, labels = CitrobacterRKImetadata$Strain_name_spot)
# matrix with intensities of peaks arranged in rows (each column is a mass-over-charge value)
```

```
IntMat <- MALDIquant::intensityMatrix(peaks)
rownames(IntMat) <- paste(CitrobacterRKImetadata$Strain_name_spot)
# remove missing values in the matrix
IntMat[is.na(IntMat)] <- 0
# normalize peaks according to the maximum intensity value for each mass spectrum
IntMat <- apply(IntMat,1,function(x) x/(max(x)))
# transpose the matrix for statistical analysis
X <- t(IntMat)
# define the known categories of mass spectra for the classification
Y <- factor(CitrobacterRKImetadata$Species)

###############################################################################
## 2. Selection of discriminant mass-over-charge values using sPLS-DA
# with 5 to 10 variables,
# up-sampling method and trained
# with the Accuracy coefficient metric

a <- MSclassifR::SelectionVar(X,
                              Y,
                              MethodSelection = c("RFERF"),
                              MethodValidation = c("cv"),
                              PreProcessing = c("center","scale","nzv","corr"),
                              NumberCV = 2,
                              Metric = "Accuracy",
                              Sizes = c(2:5),
                              Sampling = "up")

sel_moz=a$sel_moz

###############################################################################
## 3. Perform LogReg from shortlisted discriminant mass-over-charge values

# linear multinomial regression
# without sampling mehod and
# trained with the Kappa coefficient metric

model_lm=MSclassifR::LogReg(X=X,
                            moz=sel_moz,
                            Y=factor(Y),
                            number=2,
                            repeats=2,
                            Metric = "Kappa")
# Estimated model:
model_lm

# nonlinear multinomial regression using neural networks
# with up-sampling method and
# trained with the Kappa coefficient metric

model_nn=MSclassifR::LogReg(X=X,
                            moz=sel_moz,
                            Y=factor(Y),
                            number=2,
                            repeats=2,
                            kind="nnet",
                            Metric = "Kappa",
                            Sampling = "up")
# Estimated model:
model_nn

# nonlinear multinomial regression using random forests
# without down-sampling method and
# trained with the Kappa coefficient metric

model_rf=MSclassifR::LogReg(X=X,
                            moz=sel_moz,
                            Y=factor(Y),
                            number=2,
                            repeats=2,
                            kind="rf",
                            Metric = "Kappa",
                            Sampling = "down")

# Estimated model:
model_rf

# nonlinear multinomial regression using xgboost
# with down-sampling method and
# trained with the Kappa coefficient metric

model_xgb=MSclassifR::LogReg(X=X,
                             moz=sel_moz,
                             Y=factor(Y),
                             number=2,
                             repeats=2,
                             kind="xgb",
                             Metric = "Kappa",
                             Sampling = "down")
# Estimated model:
model_xgb

# nonlinear multinomial regression using svm
# with down-sampling method and
# trained with the Kappa coefficient metric

model_svm=MSclassifR::LogReg(X=X,
                             moz=sel_moz,
                             Y=factor(Y),
                             number=2,
                             repeats=2,
                             kind="svm",
                             Metric = "Kappa",
                             Sampling = "down")
# Estimated model:
model_svm
```

```
# Of note, you can also load a model already saved
# (see example in LogReg function) for the next step
###############################################################################
## 4. Probabilities of belonging to each category for the mass spectra
## and associated statitics

# Collect all the estimated models in a list

Models <- list(model_lm$train_mod,
               model_nn$train_mod,
               model_rf$train_mod,
               model_xgb$train_mod,
               model_svm$train_mod)

# Predict classes of mass spectra with 6 Da of tolerance for matching peaks.
prob_cat=MSclassifR:: PredictLogReg(peaks = peaks[c(1:5)],
                                    model = Models,
                                    moz = sel_moz,
                                    tolerance = 6,
                                    Reference = Y[c(1:5)])
```

`SelectionVar` *Variable selection using random forests, logistic regression methods or sparse partial least squares discriminant analysis (sPLS-DA).*

Description

This function performs variable selection (i.e. selection of discriminant mass-over-charge values) using either recursive feature elimination (RFE) algorithm with Random Forest, or logistic regression model, or sparse partial least squares discriminant analysis (sPLS-DA).

Usage

```
SelectionVar(X,
             Y,
             MethodSelection = c("RFERF", "RFEGlmnet", "VSURF", "sPLSDA"),
             MethodValidation = c("cv", "repeatedcv", "LOOCV"),
             PreProcessing = c("center","scale","nzv","corr"),
             Metric = c("Kappa", "Accuracy"),
             Sampling = c(NULL, "up", "down", "smote"),
             NumberCV = NULL,
             RepeatsCV = NULL,
             Sizes,
             Ntree = 1000,
             threshold = 0.01,
             ncomp.max = 10)
```

Arguments

`X`	a numeric `matrix` corresponding to a library of mass spectra. Each row of `X` is the intensities of a mass spectrum measured on mass-over-charge values.T he columns should be represented by mass-over-charge values.
`Y`	a `factor` with a length equal to the number of rows in `X` and containing the categories of each mass spectrum in `X`.
`MethodSelection`	a `charater` indicating the method used for variables selection. Four methods are avaible: recursive feature elimination (RFE) coupled with random forests (`"RFERF"`) or logistic regressions (`"RFEGlmnet"`); another method using random forests (`"VSURF"`); and a method based on sparse partial least squares discriminant analysis (`"sPLSDA"`).
`MethodValidation`	a `charater` indicating the resampling method:`"cv"` for cross-validation; `"repeatedcv"` for repeated cross-validation; and `"LOOCV"` for leave-one-out cross-validation.
`NumberCV`	a `numeric` value indicating K-folds for cross-validation. Not used for `VSURF` method.
`RepeatsCV`	a `numeric` value indication the number of repeat(s) for K-folds for cross-validation or repeated cross-validation. Not used for `VSURF` method.
`PreProcessing`	a `vector` indicating the method(s) used to pre-process the mass spectra in `X`: centering (`center`), scaling (`scale`), eliminating near zero variance predictors (`nzv`), or correlated predictors (`corr`).
`Metric`	a `character` indicating metric to select the optimal model for RFE algorithm, possibles metrics are the `"Kappa"` coefficient or `"Accuracy"`. This argument is not used for `VSURF` and `sPLSDA` methods which use the minimum mean out-of-bag (OOB) and balanced classification error rate (BER) metrics respectively.
`Sampling`	a `charater` indicating an optional subsampling method to handle imbalanced datasets: subsampling methods are either `"up"`, `"down"` or `"smote"`.
`Sizes`	a numeric `vector` indicating the number of variables to select. Not used for `VSURF` method. For the `RFERF` and `RFEGlmnet` methods, the final number of selected variables is the one giving the highest average `Metric` (`"Accruracy"` or `"Kappa"`) on the folds used for cross-validation. It is thus bounded by `NumberCV*max(Sizes)`. The `sPLSDA` method selects variables from the ones kept in latent components of the model using an automatic choice of the number of components (when the balanced classification error rate (BER) reaches a plateau).
`Ntree`	a `numeric` value indicating the number of trees in each forest, only used if `MethodSelection = "VSURF"` (1000 by default).
`ncomp.max`	a `positive Integer` indicating the maximum number of components that can be included in the sPLS-DA model (10 by default).
`threshold`	a `numeric` value corresponding to a threshold used for the optimal selection of the number of components included in the sPLS-DA model (0.01 by default). When the number of components increases and the balanced classification error rate (BER) does not change anymore, we keep the minimal number where the BER reaches a plateau (i.e. when `BER(N)-BER(N+1)<threshold`, we keep `N`). If a plateau is not reached, `ncomp.max` components are selected.

Details

See `rfe` in the `caret` R package, `VSURF` in the `VSURF` R package and `splsda` in the `mixOmics` R package for details.

For `Sampling` methods available for unbalanced data: `"up"` corresponds to the up-sampling method which consists of random sampling (with replacement) so that the minority class is the same size as the majority class; `"down"` corresponds to the down-sampling method randomly which consists of random sampling (without replacement) of the majority class so that their class frequencies match the minority class; `"smote"` corresponds to the Synthetic Minority Over sampling Technique (SMOTE) specific algorithm for data augmentation which consist of creates new data from minority class using the K Nearest Neighbor algorithm.

Value

A list composed of:

`sel_moz`	a `vector` with discriminant mass-over-chage values.

And of the results of the `rfe` function of the `caret` R package (methods `RFERF` and `RFEGlmnet`), or of the `VSURF` function of the `VSURF` R package (method `VSURF`).

For the `sPLSDA` method, it also returns the following items:

`Raw_data`	a horizontal bar plot and containing the contribution of features on each component.
`selected_variables`	`data frame` with uniques features (selected variables to keep and containing the contribution of features in order to class samples).See `plotLoadings` in the `mixOmics` R package for details.

References

Kuhn, Max. (2012). The caret Package. Journal of Statistical Software. 28.

Genuer, Robin, Jean-Michel Poggi and Christine Tuleau-Malot. VSURF : An R Package for Variable Selection Using Random Forests. R J. 7 (2015): 19.

Friedman J, Hastie T, Tibshirani R (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. Journal of Statistical Software, 33(1), 1-22.

Kim-Anh Le Cao, Florian Rohart, Ignacio Gonzalez, Sebastien Dejean with key contributors Benoit Gautier, Francois, Bartolo, contributions from Pierre Monget, Jeff Coquery, FangZou Yao and Benoit Liquet. (2016). mixOmics: Omics. Data Integration Project. R package version 6.1.1. https://CRAN.R-project.org/package=mixOmics

Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. 2002. SMOTE: synthetic minority over-sampling technique. J. Artif. Int. Res. 16, 1 (January 2002), 321–357.

Branco P, Ribeiro R, Torgo L (2016). “UBL: an R Package for Utility-Based Learning.” CoRR, abs/1604.08079.

Examples

```
library("MSclassifR")
library("MALDIquant")

###############################################################################
## 1. Pre-processing of mass spectra

# load mass spectra and their metadata
data("CitrobacterRKIspectra","CitrobacterRKImetadata", package = "MSclassifR")
# standard pre-processing of mass spectra
spectra <- MSclassifR::SignalProcessing(CitrobacterRKIspectra)
# detection of peaks in pre-processed mass spectra
peaks <- MSclassifR::PeakDetection(x = spectra, labels = CitrobacterRKImetadata$Strain_name_spot)
# matrix with intensities of peaks arranged in rows (each column is a mass-over-charge value)
IntMat <- MALDIquant::intensityMatrix(peaks)
rownames(IntMat) <- paste(CitrobacterRKImetadata$Strain_name_spot)
# remove missing values in the matrix
IntMat[is.na(IntMat)] <- 0
# normalize peaks according to the maximum intensity value for each mass spectrum
IntMat <- apply(IntMat,1,function(x) x/(max(x)))
# transpose the matrix for statistical analysis
X <- t(IntMat)
# define the known categories of mass spectra for the classification
Y <- factor(CitrobacterRKImetadata$Species)


###############################################################################
## 2. Perform variables selection using SelectionVar with RFE and random forest
# with 5 to 10 variables,
# up sampling method and trained with the Kappa coefficient metric
a <- SelectionVar(X,
                  Y,
                  MethodSelection = c("RFERF"),
                  MethodValidation = c("cv"),
                  PreProcessing = c("center","scale","nzv","corr"),
                  NumberCV = 2,
                  Metric = "Kappa",
                  Sizes = c(5:10),
                  Sampling = "up")

# Plotting peaks on the first pre-processed mass spectrum and highlighting the
# discriminant mass-over-charge values with red lines
PlotSpectra(SpectralData=spectra[[1]],Peaks=peaks[[1]],
            Peaks2=a$sel_moz,col_spec="blue",col_peak="black")

###############################################################################
## 3. Perform variables selection unsing SelectionVar with RFE
# and logistic regression (with 5 to 10 variables),
# up sampling method and trained with the Kappa metric
# It is recommended to have a large enough data set to use this method
b <- SelectionVar(X,
                  Y,
                  MethodSelection = c("RFEGlmnet"),
                  MethodValidation = c("cv"),
                  PreProcessing = c("center","scale","nzv","corr"),
                  NumberCV = 2,
                  Metric = "Kappa",
                  Sizes = c(5:10),
                  Sampling = "up")

# Plotting peaks on the first pre-processed mass spectrum and highlighting the
# discriminant mass-over-charge values with red lines
PlotSpectra(SpectralData=spectra[[1]],Peaks=peaks[[1]],
            Peaks2=b$sel_moz,col_spec="blue",col_peak="black")

################################################################################
## 4. Perform variables selection unsing sPLDA method
## (with 5 to 10 variables per components) and no sampling method
# It is recommended to have a large enough data set to use this method
c <- SelectionVar(X,
                  Y,
                  MethodSelection = c("sPLSDA"),
                  MethodValidation = c("LOOCV"),
                  PreProcessing = c("scale","nzv"),
                  Sizes = c(5:10))

# Plotting peaks on the first pre-processed mass spectrum and highlighting the
# discriminant mass-over-charge values with red lines
PlotSpectra(SpectralData=spectra[[1]],Peaks=peaks[[1]],
            Peaks2=c$sel_moz,col_spec="blue",col_peak="black")

################################################################################
## 5. Perform variables selection using SelectionVar with VSURF
# This function can last a few minutes
d <- SelectionVar(X, Y, MethodSelection = c("VSURF"))
summary(d$result)
```

`SelectionVarStat` *Variable selection using statistical tests. Estimating the number of discriminant features (mass-over-chage values).*

Description

This function performs statistical tests for each mass-over-chage value to determine which are discriminants between categories. Using the distribution of resulting p-values, it determines the expected number of discriminant features.

Usage

```
SelectionVarStat(X,
                 Y,
                 stat.test = "Limma",
                 pi0.method="abh",
                 fdr=0.05)
```

Arguments

`X`	a numeric `matrix` corresponding to a library of mass spectra. Rows of `X` are the intensities of a mass spectrum measured on mass-over-charge values. The columns are mass-over-charge values.
`Y`	a `factor` with a length equal to the number of rows in `X` and containing the categories of each mass spectrum in `X`.
`stat.test`	a `character` among "anova", "kruskal", or "Limma" (default). It corresponds to the test used to know if the intensity measured at a mass-over-charge value is significantly different between categories. "anova" is for a classical ANOVA Fisher test, "kruskal" is for the Kruskal-Wallis test, "Limma" is for an ANOVA Fisher test using the `limma` R package.
`pi0.method`	a `character` among "abh", "st.spline", "st.boot", "langaas", "histo", "pounds", "jiang", "slim". It corresponds to statistical methods used to estimate the proportion of true null hypotheses among the set of tested mass-over-charge values. See the `estim.pi0` function of th R package `cp4p` for details.
`fdr`	a `numeric` corresponding to False Discovery Rate threshold used to determine the differential mass-over-charge values. 0.05 by default.

Details

The `SelectionVarStat` function allows performing "quick" classification of mass-over-charge values. It tries to find all the mass-over-charge values (or the number of mass-over-charge values) that are discriminant between categories. This can conduct to select "correlated" (i.e. associated to intensities evolving similarly between categories) mass-over-charge values.

Value

A list composed of:

`nb_to_sel`	a `numeric` value corresponding to an estimated number of mass-over-chage values where the intensities are significantly different between categories. It depends on the statistical methods used in `pi0.method`.
`NbEstimatedPeaks`	a `vector` with discriminant mass-over-chage values resulting to the FDR threshold applied on the set of tested mass-over-charge values.

References

Gianetto, Quentin & Combes, Florence & Ramus, Claire & Bruley, Christophe & Coute, Yohann & Burger, Thomas. (2015). Technical Brief Calibration Plot for Proteomics (CP4P): A graphical tool to visually check the assumptions underlying FDR control in quantitative experiments. Proteomics. 16. 10.1002/pmic.201500189.

Examples

```
library("MSclassifR")
library("MALDIquant")

###############################################################################
## 1. Pre-processing of mass spectra

# load mass spectra and their metadata
data("CitrobacterRKIspectra","CitrobacterRKImetadata", package = "MSclassifR")
# standard pre-processing of mass spectra
spectra <- MSclassifR::SignalProcessing(CitrobacterRKIspectra)
# detection of peaks in pre-processed mass spectra
peaks <- MSclassifR::PeakDetection(x = spectra, labels = CitrobacterRKImetadata$Strain_name_spot)
# matrix with intensities of peaks arranged in rows (each column is a mass-over-charge value)
IntMat <- MALDIquant::intensityMatrix(peaks)
rownames(IntMat) <- paste(CitrobacterRKImetadata$Strain_name_spot)
# remove missing values in the matrix
IntMat[is.na(IntMat)] <- 0
# normalize peaks according to the maximum intensity value for each mass spectrum
IntMat <- apply(IntMat,1,function(x) x/(max(x)))
# transpose the matrix for statistical analysis
X <- t(IntMat)
# define the known categories of mass spectra for the classification
Y <- factor(CitrobacterRKImetadata$Species)

###############################################################################
## 2. Perform the SelectionVarStat function

OptiPeaks <- SelectionVarStat(X,
                              Y,
                              stat.test = ("Limma"),
                              pi0.method="abh",
                              fdr=0.05)

## Estimation of the number of peaks to discriminate species
OptiPeaks$nb_to_sel

## Discriminant mass-over-chage values
OptiPeaks$sel_moz
```

`SignalProcessing` *Function performing post acquisition signal processing*

Description

This function performs post acquisition signal processing for `list` of `MassSpectrum` objects using commonly used methods : transform intensities ("sqrt"), smoothing ("Wavelet"), remove baseline ("SNIP"), calibrate intensities ("TIC") and align spectra. Methods used are selected from the `MALDIquant` and `MALDIrppa` R packages.

Usage

```
SignalProcessing(x,
                 transformIntensity_method = "sqrt",
                 smoothing_method = "Wavelet",
                 removeBaseline_method = "SNIP",
                 removeBaseline_iterations = 25,
                 calibrateIntensity_method = "TIC",
                 alignSpectra_NoiseMethod = "MAD",
                 alignSpectra_method = "lowess",
                 alignSpectra_halfWs = 11,
                 alignSpectra_SN = 3,
                 referenceSpectra,
                 tolerance_align = 0.002,
                 ...)
```

Arguments

`x`	a `list` of `MassSpectrum` objects (see `MALDIquant` R package).
`transformIntensity_method`	a `character` indicating the method used to transform intensities: `"sqrt"` by default. This function can be replaced by another mathematical function such as `"log"`.
`smoothing_method`	a `character` indicating the smoothing methods used. By default, it performs undecimated `Wavelet` transform (UDWT) for `list` of `MassSpectrum` objects. This Smoothing method can be remplaced by `"SavitzkyGolay"` or `"MovingAverage"`. See `wavSmoothing` in the `MALDIrppa` R package for details.
`removeBaseline_method`	a `character` indicating the method used to remove baseline. It uses `"SNIP"` method for `list` of `MassSpectrum` objects. This baseline estimation method can be remplaced `"TopHat"`, `"ConvexHull"` or `"median"`. See `removeBaseline-methods` of the `MALDIquant` R package for details.

`removeBaseline_iterations`
: a `numeric` value indicting the number of iterations to remove baseline (by default = `25`). See `removeBaseline-methods` of the `MALDIquant` R package for details.

`calibrateIntensity_method`
: a `character` indicating the intensities calibration method used (`"TIC"` method by default). This calibration method can be remplaced by `"PQN"` or "median".See `calibrateIntensity-methods` of the `MALDIquant` R package for details.

`alignSpectra_NoiseMethod`
: a `character` indicating the noise estimation method. It uses `"MAD"` method for `list` of `MassSpectrum` objects. This noise estimation method estimation method can be remplaced `"SuperSmoother"`. See `estimateNoise-methods` of the `MALDIquant` R package for details.

`alignSpectra_method`
: a `character` indicating the warping method. It uses `"lowess"` method for `list` of `MassSpectrum` objects. This warping method method can be remplaced `"linear"`, `"quadratic"` or `"cubic"` . See `determineWarpingFunctions` of the `MALDIquant` R package for details.

`alignSpectra_halfWs`
: a `numeric` value half window size to detect peaks (by default = `11`). See `detectPeaks-methods` of the `MALDIquant` R package for details.

`alignSpectra_SN`
: a `numeric` value indicating the signal-to-noise ratio used to detect peaks (by default = `3`). See `detectPeaks-methods` of the `MALDIquant` R package for details.

`referenceSpectra`
: a `MassPeaks` reference object for alignment of the sample. See `referencePeaks` of the `MALDIquant` R package for details.

`tolerance_align`
: a `numeric` value indicating a maximal relative deviation of a peak position (mass) to be considered as identical in ppm (by default = `0.002`). See `determineWarpingFunctions` of the `MALDIquant` R package for details.

`...`
: other arguments from `MALDIrppa` packages for the `wavSmoothing` function such as `n.levels` (corresponding to the depth of the decomposiion for the wavelet function). See `wavSmoothing` of the `MALDIrppa` R package for details.

Details

The `SignalProcessing` function provides an analysis pipeline for `MassSpectrum` objects including intensity transformation, smoothing, removing baseline.

The `Wavelet` method relies on the `wavShrink` function of the `wmtsa` package and its dependencies (now archived by CRAN). The original C code by William Constantine and Keith L. Davidson, in turn including copyrighted routines by Insightful Corp., has been revised and included into `MALDIrppa` for the method to work.

All the methods used for `SignalProcessing` functions are selected from `MALDIquant` and `MALDIrppa` packages.

Value

A list of modified `MassSpectrum` objects (see `MALDIquant` R package) according to chosen arguments.

References

Gibb S, Strimmer K. MALDIquant: a versatile R package for the analysis of mass spectrometry data. Bioinformatics. 2012 Sep 1;28(17):2270-1. doi: 10.1093/bioinformatics/bts447. Epub 2012 Jul 12. PMID: 22796955.

Javier Palarea-Albaladejo, Kevin Mclean, Frank Wright, David G E Smith, MALDIrppa: quality control and robust analysis for mass spectrometry data, Bioinformatics, Volume 34, Issue 3, 01 February 2018, Pages 522 - 523, doi: 10.1093/bioinformatics/btx628

Examples

```
library("MALDIquant")
library("MSclassifR")

## Load mass spectra
data("CitrobacterRKIspectra", package = "MSclassifR")

# plot first unprocessed mass spectrum
PlotSpectra(SpectralData=CitrobacterRKIspectra[[1]], col_spec="blue")

## spectral treatment
spectra <- SignalProcessing(CitrobacterRKIspectra,
                            transformIntensity_method = "sqrt",
                            smoothing_method = "Wavelet",
                            removeBaseline_method = "SNIP",
                            removeBaseline_iterations = 25,
                            calibrateIntensity_method = "TIC",
                            alignSpectra_Method = "MAD",
                            alignSpectra_halfWs = 11,
                            alignSpectra_SN = 3,
                            tolerance_align = 0.002)


# plot first processed mass spectrum
PlotSpectra(SpectralData=spectra[[1]], col_spec="blue")
```

Index