Package 'MSclassifR'

July 2, 2025

Type Package

Title Automated Classification of Mass Spectra

Version 0.4.0

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

Description Functions to classify mass spectra in known categories, and to determine discriminant mass-to-charge values. It includes easy-to-use functions for preprocessing mass spectra, functions to determine discriminant mass-to-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-to-charge values. If you use this package in your research, please cite the associated publication (<doi:10.1016/j.eswa.2025.128796>). For a comprehensive guide, additional applications, and detailed examples of using this package, please visit our GitHub repository (<https://github.com/agodmer/MSclassifR_examples>).

URL https://github.com/agodmer/MSclassifR_examples, https://doi.org/10.1016/j.eswa.2025.128796

BugReports https://github.com/agodmer/MSclassifR_examples/issues

License GPL (>= 3)

Encoding UTF-8

LazyData true

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

Imports e1071, MALDIquantForeign, mixOmics, reshape2, ggplot2, nnet, dplyr, VSURF, metap, xgboost, glmnet, Boruta, mltools, mclust, stats, limma, car, vita, randomForest

Suggests knitr, rmarkdown,

NeedsCompilation no

RoxygenNote 7.1.1

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

Repository CRAN

Date/Publication 2025-07-02 16:20:16 UTC

Contents

`calculate_distance` *Function calculating the distance between two vectors.*

Description

This function calculates the distance between two vectors using the specified distance metric. Distance metrics available are "p-norm", "Chebyshev", "Canberra", "Overlap", "HEOM" (Heterogeneous Euclidean-Overlap Metric), "HVDM" (Heterogeneous Value Difference Metric). Used in the `fast_find_neighbors` function of our package.

Usage

```
calculate_distance(x, y, nominal_indices, p_code)
```

Arguments

`x`	Vector of numeric and/or categorical values.
`y`	Vector of numeric and/or categorical values.
`nominal_indices`	Vector indicating which positions in `x` and `y` contain categorical variables.This distinction is needed because: • For categorical variables, distances are often based on whether values match or not. • Hybrid distance metrics like HEOM and HVDM require knowing which variables are nominal to apply appropriate distance calculations.

p_code — Numeric code representing the distance metric to use:

- p >= 1: p-norm
- p = 0: Chebyshev
- p = -1: Canberra
- p = -2: Overlap (nominal attributes only)
- p = -3: HEOM (Heterogeneous Euclidean-Overlap Metric)
- p = -4: HVDM (Heterogeneous Value Difference Metric)

Details

Different distance metrics handle nominal variables differently:

- For pure numeric metrics (p >= 1, p = 0, p = -1), nominal features are ignored
- For the Overlap metric (p = -2), only nominal features are considered
- For HEOM (p = -3), numeric features use normalized Euclidean distance while nominal features use overlap distance (1 if different, 0 if same)
- For HVDM (p = -4), a specialized metric combines normalized differences for numeric features and value difference metric for nominal features

Value

A numeric value representing the distance between the two vectors.

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

`d_left_join` *Function joining two tables based not on exact matches*

Description

This function joins two tables based on a distance metric of one or more columns. It gives the same results that the `distance_left_join` function of the `fuzzyjoin` R package, but its execution time is faster. It is used to match mass spectra to short-listed mass-to-charge values in `PredictLogReg` and `PredictFastClass` functions.

Usage

```
d_left_join (x, y, by = NULL, method = "euclidean", max_dist = 1,
             distance_col = NULL)
```

Arguments

`x`	A `data.frame` object.
`y`	A `data.frame` object.
`by`	Columns by which to join the two tables. `NULL` by default (tables are joined by matching the common column names that appear in both datasets x and y).
`method`	Method to use for computing distance, either "euclidean" (default) or "manhattan".
`max_dist`	A `numeric` value indicating the maximum distance to use for joining. 1 by default.
`distance_col`	A `character` that specifies the name of a new column to be added to the output, which will contain the calculated distances between both tables. If NULL, no column is added (default).

Value

Returns a data frame that contains the results of joining the `x` dataset onto the `y` datasets, where all rows from `x` are preserved and matching data from `y` is added according to the specified criteria.

`fast_find_neighbors` *Function finding k Nearest Neighbors for each row of a matrix*

Description

This function finds the k nearest neighbors for each row in a matrix using the specified distance metric. Distance metrics available are "p-norm", "Chebyshev", "Canberra", "Overlap", "HEOM" (Heterogeneous Euclidean-Overlap Metric), "HVDM" (Heterogeneous Value Difference Metric). See the `calculate_distance` function of our package for more details on the distances.

Usage

```
fast_find_neighbors(data, nominal_indices, p_code, k)
```

Arguments

`data`	Matrix where the k nearest neighbors for each row are searched.
`nominal_indices`	Vector of column indices indicating which features are categorical (nominal) variables. This is crucial for proper distance calculation as nominal and numeric features require different handling. For example, if columns 2 and 5 contain categorical variables, nominal_indices should be `c(2, 5)`. See `calculate_distance` function.
`p_code`	Numeric code representing the distance metric to use: • p >= 1: p-norm • p = 0: Chebyshev • p = -1: Canberra • p = -2: Overlap (nominal attributes only) • p = -3: HEOM (Heterogeneous Euclidean-Overlap Metric) • p = -4: HVDM (Heterogeneous Value Difference Metric)
`k`	Number of nearest neighbors to find.

Value

A matrix where each row contains the indices of the k nearest neighbors for the corresponding example.

See Also

`calculate_distance`

`fast_generate_synthetic` *Function generating synthetic examples using SMOTE*

Description

This function generates synthetic examples using the SMOTE algorithm. For each example in the minority class, this function generates synthetic examples by interpolating between the example and its k nearest neighbors. This function is used in the `smote_classif` function.

Usage

```
fast_generate_synthetic(dat, k, n, p_code)
```

Arguments

`dat`	A `data.frame` object.
`k`	Number of nearest neighbors to consider.
`n`	Number of synthetic examples to generate.
`p_code`	Numeric code representing the distance metric to use: • p >= 1: p-norm • p = 0: Chebyshev • p = -1: Canberra • p = -2: Overlap (nominal attributes only) • p = -3: HEOM (Heterogeneous Euclidean-Overlap Metric) • p = -4: HVDM (Heterogeneous Value Difference Metric)

Value

A data frame containing the generated synthetic examples.

See Also

`smote_classif`

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

Description

This function estimates a multinomial regression using cross-validation to predict the category (species, phenotypes...) to which a mass spectrum belongs from a set of shortlisted mass-to-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("no", "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-to-charge values.
moz	vector with shortlisted mass-to-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 character indicating metric to select the optimal model. The possibles metrics are the "Kappa" coefficient,"Accuracy", 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 `character` indicating an optional subsampling method to handle imbalanced datasets: subsampling methods are either `"no"` (no subsampling), `"up"`, `"down"` or `"smote"`. `"no"` by default.

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:

$$Accuracy = NumberOfCorrectPredictions/TotalNumberOfPredictions$$

$$Kappacoefficient = (ObservedAgreement - ChanceAgreement)/(1 - ChanceAgreement)$$

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

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

The `Sampling` methods available for imbalanced data are: `"up"` 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"` to the down-sampling method which consists of random sampling (without replacement) of the majority class so that their class frequencies match the minority class; `"smote"` to the Synthetic Minority Over sampling Technique (SMOTE) algorithm for data augmentation which consists of creating new data from minority class using the K Nearest Neighbor algorithm.

Godmer et al. (2025) presents a comparison of different pipelines using LogReg that can help you to optimize your workflow. For a comprehensive guide, additional applications, and detailed examples of using this package, please visit our GitHub repository: here.

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.

Alexandre Godmer, Yahia Benzerara, Emmanuelle Varon, Nicolas Veziris, Karen Druart, Renaud Mozet, Mariette Matondo, Alexandra Aubry, Quentin Giai Gianetto, MSclassifR: An R package for supervised classification of mass spectra with machine learning methods, Expert Systems with Applications, Volume 294, 2025, 128796, ISSN 0957-4174, doi:10.1016/j.eswa.2025.128796.

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 <- MSclassifR::PeakDetection(x = spectra, averageMassSpec=FALSE)
# matrix with intensities of peaks arranged in rows (each column is a mass-to-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-to-charge values using RFERF
# 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-to-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-to-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-to-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-to-charge values.

If you use this package in your research, please cite the associated publication: doi:10.1016/j.eswa.2025.128796.

For a comprehensive guide, additional applications, and detailed examples of using this package, please visit our GitHub repository: here.

Value

No return value. Package description.

Author(s)

Alexandre Godmer, Quentin Giai Gianetto

References

Alexandre Godmer, Yahia Benzerara, Emmanuelle Varon, Nicolas Veziris, Karen Druart, Renaud Mozet, Mariette Matondo, Alexandra Aubry, Quentin Giai Gianetto, MSclassifR: An R package for supervised classification of mass spectra with machine learning methods, Expert Systems with Applications, Volume 294, 2025, 128796, ISSN 0957-4174, doi:10.1016/j.eswa.2025.128796.

`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,
              averageMassSpec = TRUE,
              labels = NULL,
              averageMassSpectraMethod = "median",
              SNRdetection = 3,
              binPeaks = TRUE,
              PeakDetectionMethod = "MAD",
              halfWindowSizeDetection = 11,
              AlignMethod = "strict",
              Tolerance = 0.002,
              ...)
```

Arguments

`x`	a `list` of `MassSpectrum` objects (see `MALDIquant` R package).
`averageMassSpec`	a `logical` value indicating whether it is necessary to group the spectra according to the arguments `labels` and `averageMassSpectraMethod`. It is fixed to `"TRUE"` by default.
`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.

binPeaks
: a logical value indicating the peaks are aligned in discrete bins. It is fixed to "TRUE" by default. See binPeaks 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

Alexandre Godmer, Yahia Benzerara, Emmanuelle Varon, Nicolas Veziris, Karen Druart, Renaud Mozet, Mariette Matondo, Alexandra Aubry, Quentin Giai Gianetto, MSclassifR: An R package for supervised classification of mass spectra with machine learning methods, Expert Systems with Applications, Volume 294, 2025, 128796, ISSN 0957-4174, doi:10.1016/j.eswa.2025.128796.

Examples

```
library("MALDIquant")
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,
                       averageMassSpec = FALSE,
                       labels = CitrobacterRKImetadata$Strain_name_spot,
                       averageMassSpectraMethod = "median",
                       SNRdetection = 3,
                       binPeaks = TRUE,
                       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-to-charge values to plot.
`Peaks`	`MassPeaks` object (see `MALDIquant` R package). If NULL, peaks are not highlighted.
`Peaks2`	numeric `vector` of mass-to-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-to-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, averageMassSpec=FALSE)
# 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-to-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-to-charge values that corresponds to mass-to-charge values in the columns of mod_peaks. By default, all the mass-to-charge values in mod_peaks are used.
tolerance	a numeric value of accepted tolerance to match peaks to the set of shortlisted mass-to-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-to-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-to-charge values moz. It is fixed to 0 by default.

Details

Godmer et al. (2025) presents a comparison of different pipelines using PredictFastClass that can help you to optimize your workflow. For a comprehensive guide, additional applications, and detailed examples of using this package, please visit our GitHub repository: here.

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.

References

Alexandre Godmer, Yahia Benzerara, Emmanuelle Varon, Nicolas Veziris, Karen Druart, Renaud Mozet, Mariette Matondo, Alexandra Aubry, Quentin Giai Gianetto, MSclassifR: An R package for supervised classification of mass spectra with machine learning methods, Expert Systems with Applications, Volume 294, 2025, 128796, ISSN 0957-4174, doi:10.1016/j.eswa.2025.128796.

Examples

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

# 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 <- peaks <- MSclassifR::PeakDetection(x = spectra, averageMassSpec=FALSE)
# matrix with intensities of peaks arranged in rows (each column is a mass-to-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)

#Predict species without peak selection using a tolerance of 1 Da
res = PredictFastClass(peaks=peaks[1:5],
                       mod_peaks=X,
                       Y_mod_peaks=Y,
                       tolerance = 1)

#comparing predicted categories (species) and the truth
cbind(res$pred_cat,as.character(Y[1:5]))

# The method can be applied after a peak selection step
a <- SelectionVar(X,
                  Y,
                  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 1 Da
res = PredictFastClass(peaks=peaks[1:5],
                       moz = a$sel_moz,
                       mod_peaks=X,
                       Y_mod_peaks=Y, tolerance = 1)

#comparing predicted categories (species) and the truth
cbind(res$pred_cat,as.character(Y[1:5]))
```

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-to-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-to-charge values (output of the LogReg function).
moz	a vector with the set of shortlisted mass-to-charge values used to estimate the model Model.
tolerance	a numeric value of accepted tolerance to match peaks to the set of shortlisted mass-to-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-to-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-to-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 predicting the membership of a mass spectrum to a category from a multinomial regression model. The mass spectrum from the peaks object will be matched to the discriminant mass-to-chage (m/z) values (sel_moz object from the SelectionVar or SelectionVarStat functions) with a tolerance between 2 m/z and defined by the tolerance parameter (by default this value is 6 Da). If a repetition of a 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.

Godmer et al. (2025) presents a comparison of different pipelines using PredictLogReg that can help you to optimize your workflow. For a comprehensive guide, additional applications, and detailed examples of using this package, please visit our GitHub repository: here.

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.

Alexandre Godmer, Yahia Benzerara, Emmanuelle Varon, Nicolas Veziris, Karen Druart, Renaud Mozet, Mariette Matondo, Alexandra Aubry, Quentin Giai Gianetto, MSclassifR: An R package for supervised classification of mass spectra with machine learning methods, Expert Systems with Applications, Volume 294, 2025, 128796, ISSN 0957-4174, doi:10.1016/j.eswa.2025.128796.

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 <- MSclassifR::PeakDetection(x = spectra, averageMassSpec=FALSE)
# matrix with intensities of peaks arranged in rows (each column is a mass-to-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-to-charge values using RFERF
# with 5 to 10 variables,
# without 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 = "Kappa",
                              Sizes = c(5:10))

sel_moz=a$sel_moz
```

```
################################################################################
## 3. Perform LogReg from shortlisted discriminant mass-to-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)])

##############################################################################
## 5. Example of meta-classifiers based on several random forest models
## to optimize a Kappa value using the SMOTE method for imbalanced datasets.
## -> a merge of the prediction probabilities using the Fisher's method
## leads generally to robust prediction models.

#Selecting peaks with mda method
```

```
a=SelectionVar(X,Y,MethodSelection="mda",Ntree=5*ncol(X))
sel_moz=a$sel_moz

#Building 4 Random Forest models
models=NULL;nbm=4;
for (i in 1:nbm){
  model_rf=MSclassifR::LogReg(X=X,
                              moz=sel_moz,
                              Y=factor(Y),
                              number=5,
                              repeats=5,
                              kind="rf",
                              Metric = "Kappa",
                              Sampling = "smote")
  models <- c(models,list(model_rf$train_mod))
}

#Combining their prediction probabilities
prob_cat=MSclassifR::PredictLogReg(peaks = peaks,model = models,moz = sel_moz,
tolerance = 6,Reference = Y)
```

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-to-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) or methods based on the distribution of variable importances of random forests.

Usage

```
SelectionVar(X,
             Y,
            MethodSelection = c("RFERF", "RFEGlmnet", "VSURF", "sPLSDA", "mda",
                                "cvp", "boruta"),
             MethodValidation = c("cv", "repeatedcv", "LOOCV"),
             PreProcessing = c("center", "scale", "nzv", "corr"),
             Metric = c("Kappa", "Accuracy"),
             Sampling = c("no", "up","down", "smote"),
             NumberCV = NULL,
             RepeatsCV = NULL,
             Sizes,
             Ntree = 1000,
             ncores = 2,
```

```
                threshold = 0.01,
                ncomp.max = 10,
                nbf=0)
```

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-to-charge values. The columns are assumed to be mass-to-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 character indicating the method used for variables selection. Methods available: (1) "RFERF" for recursive feature elimination (RFE) coupled with random forests (see rfe in the caret R package); (2) "RFEGlmnet" for RFE with coupled with logistic regression; (3) "VSURF" for a method using random forests (see VSURF in the VSURF R package); (4) "sPLSDA" for a method based on sparse partial least squares discriminant analysis (see splsda in the mixOmics); (5) "mda" for a method selecting variables from the distribution of the "mean decrease in accuracy" variables importances of a random forest (see importance function in the randomForest R package); (6) "cvp" for a method selecting variables from the distribution of the cross-validated permutation variables importances of a random forest (see CVPVI function in the vita R package); (7) "boruta" for a method selecting variables using the Boruta algorithm that iteratively compares importances of variables with importances of shadow variables, created by shuffling original ones (see Boruta function in the Boruta R package). Additional explanations are available in the Details section.
MethodValidation	a character indicating the resampling method:"cv" for cross-validation; "repeatedcv" for repeated cross-validation; and "LOOCV" for leave-one-out cross-validation. Only used for the "RFERF", "RFEGlmnet" and "sPLSDA" methods.
NumberCV	a numeric value indicating the number of K-folds for cross-validation. Only used for the "RFERF", "RFEGlmnet", "sPLSDA" and "cvp" methods.
RepeatsCV	a numeric value indication the number of repeat(s) for K-folds for cross-validation or repeated cross-validation. Only used for the "RFERF", "RFEGlmnet" and "sPLSDA" methods.
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"). Only used for the "RFERF", "RFEGlmnet" and "sPLSDA" methods.
Metric	a character indicating the metric used to select the optimal model for the RFE algorithms. Possible metrics are the "Kappa" coefficient or the "Accuracy". This argument is not used for the "VSURF", "cvp", "mda" and the "sPLSDA" methods of MethodSelection. See details of the "SelectionVar" function.
Sampling	a character indicating an optional subsampling method to handle imbalanced datasets: subsampling methods are either "no" (no subsampling), "up", "down" or "smote". "no" by default.

`Sizes` a numeric vector indicating the number of variables to select. Only used for the "RFERF", "RFEGlmnet" and "sPLSDA" methods. For the "RFERF" and "RFEGlmnet" methods, the final number of selected variables is the one giving the highest average "Metric" ("Accuracy" or "Kappa") on the folds used for cross-validation. It is thus bounded by NumberCV*max(Sizes). For the "sPLSDA" method, Sizes corresponds to the number of variables to test from the X dataset when estimating the sparse PLS-DA model (see test.keepX argument in the mixOmics R package).

`Ntree` a numeric value indicating the number of trees in random forests, only used if MethodSelection = "VSURF" or "mda" or "cvp". Note we advise to select a number highly superior to the total number of variables for a robust selection (to not miss some features in the subspaces used to build trees). It is 1000 by default.

`ncores` a positive integer only used for the cvp method. The number of cores to use, i.e. at most how many child processes will be run simultaneously. Must be at least one, and parallelization requires at least two cores. If ncores=0, then the half of CPU cores on the current host are used.

`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.

`nbf` a numeric value corresponding to a number of simulated non discriminant features. This is used to improve the robustness of the estimation of the distribution of the variable importances for non discriminant features. Only used for the "mda" and "cvp" methods. 0 by default: no additional non discriminant feature is created.

Details

The selection of variables can be carried out with two different objectives: either to find a minimum number of variables allowing to obtain the highest possible accuracy (or Kappa coefficient), which involves the possible elimination of variables correlated between them (i.e. not bringing any additional predictive power with respect to some other variables); or to find all the variables in the dataset with a potential predictive power ("discriminant" variables).

The VSURF method attempts to accomplish only the first objective. The mda and cvp methods attempt to accomplish the second objective, as do the methods available in the SelectionVarStat function of our MSclassifR R package. The RFERF, RFEGlmnet and sPLSDA methods take as input a number of variables to be selected(Sizes argument), and can therefore be used with both objectives.

Within the framework of the second objective, either the mda or cvp methods can be used to estimate a number of discriminant variables from the importances of variables. The SelectionVarStat function can also be used to estimate this number from distributions of p-values. Of note, be sure that the Ntree argument is high enough to get a robust estimation with the mda or cvp methods.

The "RFEGlmnet" and "RFERF" methods are based on recursive feature elimination and can either optimize the kappa coefficient or the accuracy as metrics when selecting variables.

The "sPLSDA" method selects variables from the ones kept in latent components of the sparse PLS-DA model using an automatic choice of the number of components (when the balanced classification error rate (BER) reaches a plateau - see argument threshold).

The "mda" and "cvp" methods use the distribution of variable importances to estimate the number of discriminant features (mass-to-charge values). Briefly, the distribution of variable importances for useless (not discriminant) features is firstly estimated from negative importance variables by the method proposed in section 2.6 of Janitza et al.(2018). Next, the following mixture model is assumed: $F(x) = \pi \times F_u(x) + (1 - \pi) \times F_d(x)$ where F is the empirical cumulative distribution of variable importances of all the features, F_u the one of the useless features, F_d the one of the discriminative features, and π is the proportion of useless features in the dataset. From the estimated distribution of useless features, we can estimate quantile values x_q and compute $\epsilon_q = min(F(x_q)/q; 1)$ for each quantile q. The minimum of the ϵ_q corresponds to the estimated proportion of useless features in the dataset, what allows estimating the number of discriminant features by $N_d = floor(N \times (1 - \pi))$ where N is the total number of features. Next, the N_d features with the highest variable importances are selected.

The "VSURF" and "sPLSDA" methods use the minimum mean out-of-bag (OOB) and balanced classification error rate (BER) metrics respectively.

The "boruta" method selects variables from the Boruta algorithm (see Kursa and Rudnicki (2010)). The maxRuns argument of the Boruta function is fixed to 3*ncol(X) to perform the selection of variables.

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.

See rfe in the caret R package, VSURF in the VSURF R package, splsda in the mixOmics R package, importance function in the randomForest R package, CVPVI function in the vita R package, and Boruta function in the Boruta R package for more details.

Godmer et al. (2025) presents a comparison of different pipelines using SelectionVar that can help you to optimize your workflow. For a comprehensive guide, additional applications, and detailed examples of using this package, please visit our GitHub repository: here.

Value

A list composed of:

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

For the "RFERF" and "RFEGlmnet" methods, it also returns the results of the rfe function of the caret R package.

For the "VSURF" method, it also returns the results of the results of the VSURF function of the VSURF R package.

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.

For the `"mda"` and `"cvp"` methods, it also returns the following items:

`nb_to_sel` a numeric value corresponding to an estimated number of mass-over-chage values where the intensities are significantly different between categories (see details).

`imp_sel` a vector containing the variable importances for the selected features.

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.

Janitza, S., Celik, E., Boulesteix, A. L. (2018). A computationally fast variable importance test for random forests for high-dimensional data. Advances in Data Analysis and Classification, 12, 885-915.

Miron B. Kursa, Witold R. Rudnicki (2010). Feature Selection with the Boruta Package. Journal of Statistical Software, 36(11), p. 1-13.

Alexandre Godmer, Yahia Benzerara, Emmanuelle Varon, Nicolas Veziris, Karen Druart, Renaud Mozet, Mariette Matondo, Alexandra Aubry, Quentin Giai Gianetto, MSclassifR: An R package for supervised classification of mass spectra with machine learning methods, Expert Systems with Applications, Volume 294, 2025, 128796, ISSN 0957-4174, doi:10.1016/j.eswa.2025.128796.

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, averageMassSpec=FALSE)
# matrix with intensities of peaks arranged in rows (each column is a mass-to-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-to-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 using SelectionVar with VSURF
# This function can last a few minutes
b <- SelectionVar(X, Y, MethodSelection = c("VSURF"))
summary(b$result)

##############################################################################
## 4. Perform variables selection using SelectionVar with "mda" or "cvp"
# option 1: Using mean decrease in accuracy
# with no sampling method
c <- SelectionVar(X,Y,MethodSelection="mda",Ntree=10*ncol(X))

# Estimation of the number of peaks to discriminate species
c$nb_to_sel
```

```
# Discriminant mass-to-charge values
c$sel_moz

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

# option 2: Using cross-validated permutation variable importance measures (more "time-consuming")
# with no sampling method
d <- SelectionVar(X,Y,MethodSelection="cvp",NumberCV=2,ncores=2,Ntree=1000)

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

# Discriminant mass-to-charge values
d$sel_moz

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

# Mass-over charge values found with both methods ("mda" and "cvp")
intersect(c$sel_moz,d$sel_moz)
```

`SelectionVarStat` *Variable selection using multiple statistical tests.*

Description

This function performs a statistical test for each mass-to-charge value to determine which are discriminants between categories. Using the distribution of resulting multiple p-values, it determines an expected number of discriminant features, and adjusted p-values that can be used to control a false discovery rate threshold.

Usage

```
SelectionVarStat(X,
                 Y,
                 stat.test = "Limma",
                 pi0.method="abh",
                 fdr=0.05,
                 Sampling = c("no", "up","down", "smote"))
```

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-to-charge values. The columns are mass-to-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-to-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-to-charge values. See the estim.pi0 function of the R package cp4p for details.
fdr	a numeric value corresponding to False Discovery Rate threshold used to determine the differential mass-to-charge values. 0.05 by default.
Sampling	a character indicating an optional subsampling method to handle imbalanced datasets: subsampling methods are either "no" (no subsampling), "up", "down" or "smote". "no" by default.

Details

The SelectionVarStat function allows performing "quick" classification of mass-to-charge values. It tries to find all the mass-to-charge values (or the number of mass-to-charge values) that are discriminant between categories. This can conduct to select "correlated" mass-to-charge values (i.e. associated to intensities evolving similarly between categories). By default, multiple moderated t-tests using the limma R package (bayesian regularization of variances) are performed and the p-values are corrected using an adaptive Benjamini and Hochberg procedure to control the false discovery rate. Different ways to estimate the proportion of true null hypotheses (object pi0 returned by the function - see the cp4p R package for details) can be used for the adaptive Benjamini-Hochberg procedure ("abh" by defaut).

Godmer et al. (2025) presents a comparison of different pipelines using SelectionVarStat that can help you to optimize your workflow. For a comprehensive guide, additional applications, and detailed examples of using this package, please visit our GitHub repository: here.

Value

A list composed of:

nb_to_sel	a numeric value corresponding to an estimation of the optimal number of mass-to-charge values to discriminate between different groups.
sel_moz	a vector with selected discriminant mass-to-charge values.
ap	a list composed of pi0 the proportion of non-discriminant mass-to-charge values, and adjp a matrix of raw p-values and corresponding ajusted p-values for all the mass-to-charge values that have been tested.

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.

Alexandre Godmer, Yahia Benzerara, Emmanuelle Varon, Nicolas Veziris, Karen Druart, Renaud Mozet, Mariette Matondo, Alexandra Aubry, Quentin Giai Gianetto, MSclassifR: An R package for supervised classification of mass spectra with machine learning methods, Expert Systems with Applications, Volume 294, 2025, 128796, ISSN 0957-4174, doi:10.1016/j.eswa.2025.128796.

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$Strain_name_spot)
# matrix with intensities of peaks arranged in rows (each column is a mass-to-charge value)
IntMat <- MALDIquant::intensityMatrix(peaks)
rownames(IntMat) <- paste(CitrobacterRKImetadata$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. Estimate the optimal number of peaks to discriminate the different species

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

## Estimation of the optimal number of peaks to discriminate species (from the pi0 parameter)
OptiPeaks$nb_to_sel

## discriminant mass-to-chage values estimated using a 5 per cent false discovery rate
OptiPeaks$sel_moz
```

```
## p-values and adjusted p-values estimated for all the tested mass-to-charge values
OptiPeaks$ap$adjp
```

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,
                 tolerance_align = 0.002,
                 referenceSpectra = NULL,
                 minFrequency= 0.5,
                 binPeaks_method = "strict",
                 keepReferenceSpectra = FALSE,
                 ...)
```

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.

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.

referenceSpectra
: a MassPeaks reference spectrum for alignment of the sample(s). If this reference spectrum is not provided, a reference spectrum is created using the minFrequency and binPeaks_method arguments. See referencePeaks of the MALDIquant R package for details.

minFrequency
: a numeric minimum frequency for each peak over all analyzed spectra (by default = 0.5) for the creation of the reference spectrum. See referencePeaks of the MALDIquant R package for details.

binPeaks_method
: a character indicating the method used to equalize masses for similar peaks for the creation of the reference spectrum. 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.

keepReferenceSpectra
: a logical value indicating if the created reference spectrum is returned by the function (FALSE, default).

...
: 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.

Godmer et al. (2025) presents a comparison of different pre-processing workflows that can help you to optimize your workflow.For a comprehensive guide, additional applications, and detailed examples of using this package, please visit our GitHub repository: here.

Value

A list of modified MassSpectrum objects (see MALDIquant R package) according to chosen arguments. If the argument referenceSpectra is not completed and the argument keepReferenceSpectra is TRUE, a list containing the MassSpectrum objects modified named "spectra" and the created reference spectrum named "RefS" is returned.

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

Alexandre Godmer, Yahia Benzerara, Emmanuelle Varon, Nicolas Veziris, Karen Druart, Renaud Mozet, Mariette Matondo, Alexandra Aubry, Quentin Giai Gianetto, MSclassifR: An R package for supervised classification of mass spectra with machine learning methods, Expert Systems with Applications, Volume 294, 2025, 128796, ISSN 0957-4174, doi:10.1016/j.eswa.2025.128796.

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")
```

`smote_classif` *SMOTE for Classification Problems*

Description

This function performs Synthetic Minority Over-sampling Technique (SMOTE) to address class imbalance in classification problems. This implementation supports various distance metrics, different balancing strategies, and handles mixed data types (numeric and categorical). Its execution time is faster than the `SmoteClassif` of the `UBL` R package.

Usage

```
smote_classif (form, dat, C.perc = "balance", k = 5, repl = FALSE,
                           dist = "Euclidean", p = 2)
```

Arguments

`form`	A model formula identifying the target variable (e.g., `Class ~ .`).
`dat`	A data frame containing the imbalanced dataset.
`C.perc`	Either `"balance"`, `"extreme"`, or a named list containing over/under-sampling percentages for each class. Values < 1 indicate under-sampling, values > 1 indicate over-sampling, and values = 1 indicate no change. "balance" equalizes all classes, `"extreme"` performs more aggressive balancing.
`k`	Integer specifying the number of nearest neighbors to use when generating synthetic examples (default: 5).

repl	Logical, whether to allow sampling with replacement when under-sampling (default: FALSE).
dist	Distance metric to use for nearest neighbor calculations. Supported metrics: "Euclidean" (default), "Manhattan", "Chebyshev", "Canberra", "Overlap", "HEOM", "HVDM", or "p-norm". See calculate_distance function for details.
p	Parameter used when dist = "p-norm" (default: 2).

Details

If you use this package in your research, please cite the associated publication (doi:10.1016/j.eswa.2025.128796).

Value

A data frame with the same structure as the input, but with rebalanced classes according to the specified strategy.

References

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. Journal of Artificial Intelligence Research, 16, 321-357.doi:10.1613/jair.953.

Alexandre Godmer, Yahia Benzerara, Emmanuelle Varon, Nicolas Veziris, Karen Druart, Renaud Mozet, Mariette Matondo, Alexandra Aubry, Quentin Giai Gianetto, MSclassifR: An R package for supervised classification of mass spectra with machine learning methods, Expert Systems with Applications, Volume 294, 2025, 128796, ISSN 0957-4174, doi:10.1016/j.eswa.2025.128796.

See Also

fast_generate_synthetic for the implementation of synthetic example generation

Examples

```
# Load the iris dataset
data(iris)

# Create an imbalanced dataset by taking a subset
imbal_iris <- iris[c(1:40, 51:100, 101:110), ]
table(imbal_iris$Species)  # Show class distribution

# Balance classes using the default "balance" strategy
balanced_iris <- smote_classif(Species ~ ., imbal_iris)
table(balanced_iris$Species)  # Show balanced distribution

# Custom over/under-sampling
custom_iris <- smote_classif(Species ~ ., imbal_iris,
                             C.perc = list(setosa = 0.8,
                                           versicolor = 1,
                                           virginica = 3))
table(custom_iris$Species)
```

Index