Package 'ASICS'

December 30, 2024

Type Package

Title Automatic Statistical Identification in Complex Spectra

Version 2.23.0

Description With a set of pure metabolite reference spectra, ASICS quantifies concentration of metabolites in a complex spectrum. The identification of metabolites is performed by fitting a mixture model to the spectra of the library with a sparse penalty. The method and its statistical properties are described in Tardivel et al. (2017) <doi:10.1007/s11306-017-1244-5>.

Depends R (>= 3.5)

Imports BiocParallel, ggplot2, glmnet, grDevices, gridExtra, methods, mvtnorm, PepsNMR, plyr, quadprog, ropls, stats, SummarizedExperiment, utils, Matrix, zoo

License GPL (>= 2)

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Suggests knitr, rmarkdown, BiocStyle, testthat, ASICSdata

VignetteBuilder knitr

Collate 'ASICS.R' 'Spectra-class.R' 'ASICSResults-class.R' 'AnalysisResults-class.R' 'PureLibrary-class.R' 'alignment.R' 'analysis_on_quantification.R' 'concentration_optimisation.R' 'data.R' 'library_transformation.R' 'load_data.R' 'plotAlignment.R' 'plot_spectrum.R' 'simulation.R' 'user_guide.R' 'utils.R'

biocViews Software, DataImport, Cheminformatics, Metabolomics

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

git_branch devel

git_last_commit 034bc20

git_last_commit_date 2024-10-29

Repository Bioconductor 3.21

Date/Publication 2024-12-30

Author Gaëlle Lefort [aut, cre],
Rémi Servien [aut],
Patrick Tardivel [aut],
Nathalie Vialaneix [aut]

Maintainer Gaëlle Lefort <gaelle.lefort@inrae.fr>

Contents

`accessors-methods` *Accessors*

Description

List of available accessors for each slot of all S4 classes present in the package.

Usage

```
## S4 method for signature 'Spectra'
getSampleName(object)

## S4 method for signature 'Spectra'
getPpmGrid(object)

## S4 method for signature 'Spectra'
getSpectra(object)

## S4 method for signature 'Spectra'
getNormMethod(object)

## S4 method for signature 'Spectra'
getNormParams(object)

## S4 method for signature 'ASICSResults'
getReconstructedSpectra(object)

## S4 method for signature 'ASICSResults'
getQuantification(object)

## S4 method for signature 'ASICSResults'
getDeformedLibrary(object)

## S4 method for signature 'AnalysisResults'
getTypeAnalysis(object)

## S4 method for signature 'AnalysisResults'
getTypeData(object)

## S4 method for signature 'AnalysisResults'
getDataset(object)

## S4 method for signature 'AnalysisResults'
getResults(object)

## S4 method for signature 'AnalysisResults'
getBestModel(object)

## S4 method for signature 'AnalysisResults'
getCVError(object)

## S4 method for signature 'AnalysisResults'
getMeanByGroup(object)

## S4 method for signature 'PureLibrary'
getNbProtons(object)
```

Arguments

object	An object of class Spectra, PureLibrary, ASICSResults or AnalysisResults.

Value

The wanted accessor

Examples

```
# Import data and create object
current_path <- file.path(system.file("extdata", package = "ASICS"),
                          "spectra_example.txt")
spectra_data <- read.table(current_path, header = TRUE, row.names = 1)
spectra_obj <- createSpectra(spectra_data)

# Sample names
getSampleName(spectra_obj)
# Spectra
getSpectra(spectra_obj)
```

---

alignSpectra *Alignment*

Description

Align spectra of a data frame by a method based on the CluPA algorithm (Vu et al., (2011))

Usage

```
alignSpectra(
  spectra,
  reference = NULL,
  max.shift = 0.02,
  ncores = 1,
  verbose = TRUE
)
```

Arguments

spectra	Data frame with spectra in columns and chemical shift in rows. Colnames of this data frame correspond to pure metabolite names and rownames to chemical shift grid (in ppm).
reference	Index of the reference spectrum used for the alignment. Default to NULL, *i.e.* the reference spectrum is automatically detected.
max.shift	Maximum shift allowed for the alignment. Default to 0.002.
ncores	Number of cores used in parallel evaluation. Default to 1.
verbose	A boolean value to allow print out process information.

Value

A data frame with aligned spectra in columns and chemical shifts (in ppm) in rows.

References

Vu, T. N., Valkenborg, D., Smets, K., Verwaest, K. A., Dommisse, R., Lemiere, F., ... & Laukens, K. (2011). An integrated workflow for robust alignment and simplified quantitative analysis of NMR spectrometry data. *BMC Bioinformatics*, **12**(1), 405.

Examples

```
current_path <- system.file("extdata", package = "ASICS")
spectra_data <- importSpectra(name.dir = current_path,
                     name.file = "spectra_example.txt", type.import = "txt")
spectra_align <- alignSpectra(spectra_data)
```

AnalysisResults-class *Class AnalysisResults*

Description

Objects of class AnalysisResults contains results of analyses performed with the functions `pca`, `oplsda` and `kruskalWallis`.

Slots

`type.analysis` Name of the analysis (*e.g.,* `"PCA"`, `"OPLS-DA"`, ...).

`type.data` Type of data used for the analyses (*e.g.,* `"quantification"`, `"buckets"`...).

`dataset` The object of type `SummarizedExperiment` used for the analysis.

`results` Results of the analysis. Can be a data frame for test results or an object of class `opls` from `ropls` for PCA and OPLS-DA.

`best.model` Best model (only for OPLS-DA analyses).

`cv.error` Cross validation error (only for OPLS-DA analyses).

`mean.by.group` Data frame with means by group and a variable indicating if there is a significant difference between groups for tests and if the VIP associated to the variable is superior to the given threshold for OPLS-DA.

Methods

Multiple methods can be applied on AnalysisResults objects.

- As usual for S4 object, show and summary methods are available, see Object summary
- All slots have an accessor `get_slot name`, see Accessors
- All results contained in an object can be represent in a plot, see Visualisation methods

ASICS *Automatic Statistical Identification in Complex Spectra*

Description

Quantification of 1D 1H NMR spectra with ASICS method using a library of pure metabolite spectra. The method is presented in Tardivel et al. (2017).

Usage

```
ASICS(
  spectra_obj,
  exclusion.areas = matrix(c(4.5, 5.1), ncol = 2),
  max.shift = 0.02,
  pure.library = NULL,
  noise.thres = 0.02,
  joint.align = TRUE,
  threshold.noise = NULL,
  combine = NULL,
  add.noise = 0.15,
  mult.noise = 0.172,
  quantif.method = c("FWER", "Lasso", "both"),
  clean.thres = 1,
  ref.spectrum = NULL,
  seed = 1234,
  ncores = 1,
  verbose = TRUE
)
```

Arguments

`spectra_obj`	An object of class Spectra obtained with the function createSpectra.
`exclusion.areas`	Definition domain of spectra that has to be excluded for the quantification (ppm). By default, the water region is excluded (4.5-5.1 ppm).
`max.shift`	Maximum chemical shift allowed (in ppm). Default to 0.02.
`pure.library`	An object of class PureLibrary containing the reference spectra (pure metabolite spectra). If `NULL`, the library included in the package (that contains 191 reference spectra) is used.
`noise.thres`	Threshold for signal noise. Default to 0.02.
`joint.align`	Logical. If `TRUE`, information from all spectra is taken into account to align individual library.
`threshold.noise`	DEPRECATED, use `noise.thres` instead.
`combine`	DEPRECATED, use `joint.align` instead.

`add.noise`, `mult.noise`
: additive and multiplicative noises. To set these noises, you can compute the standard deviation in a noisy area for `add.noise` or the standard deviation in a peak area for `mult.noise` when several spectra of the same sample are available. By default, `add.noise` = `0.15` and `mult.noise` = `0.172`

`quantif.method`
: either `"FWER"` to perform an independent quantification (the method available in ASICS since the beginning), `"Lasso"` to perform a joint quantification (all the spectra together) or `"both"` to perform a joint quantification after the FWER selection of the independent quantification. More details can be founded in the user's guide.

`clean.thres`
: if `quantif.method == "both"` the percentage of spectra in which the metabolite needs to be identified by the FWER selection. Default to 1, *i.e.* metabolite is quantified if it was identified in at least 1% of the spectra.

`ref.spectrum`
: index of the reference spectrum used for the alignment. Default to `NULL`, *i.e.* the reference spectrum is automatically detected.

`seed`
: Random seed to control randomness in the algorithm (used in the estimation of the significativity of a given metabolite concentration).

`ncores`
: Number of cores used in parallel evaluation. Default to `1`.

`verbose`
: A Boolean value to allow print out process information.

Value

An object of type ASICSResults containing the quantification results.

Note

Since version 2.3.1 small changes were applied in order to improve the speed of metabolite selection algorithm, which can slightly impact outputs of the method.

References

Tardivel P., Canlet C., Lefort G., Tremblay-Franco M., Debrauwer L., Concordet D., Servien R. (2017). ASICS: an automatic method for identification and quantification of metabolites in complex 1D 1H NMR spectra. *Metabolomics*, **13**(10): 109. https://doi.org/10.1007/s11306-017-1244-5

See Also

ASICSResults `pure_library` `createSpectra`

Examples

```
# Import data and create object
current_path <- system.file("extdata", package = "ASICS")
spectra_data <- importSpectra(name.dir = current_path,
                     name.file = "spectra_example.txt", type.import = "txt")
spectra_obj <- createSpectra(spectra_data)

# Estimation of relative quantifications
to_exclude <- matrix(c(4.5, 10), ncol = 2)
```

```
resASICS <- ASICS(spectra_obj, exclusion.areas = to_exclude,
                  joint.align = FALSE, quantif.method = "FWER")
```

`ASICSResults-class` *Class ASICSResults*

Description

Objects of class ASICSResults contains results of ASICS quantification method for a set of spectra. This object is an extension of the class Spectra, with additional slots for quantification results, reconstructed spectra and deformed library.

Slots

`sample.name` Character vector of sample names.

`ppm.grid` Numeric vector of a unique grid (definition domain) for all spectra (in ppm).

`spectra` Numeric matrix of original spectra. Columns contain the spectra and are in the same order than `sample.name`. Rows correspond to points of `ppm.grid`.

`reconstructed.spectra` Numeric matrix of reconstructed spectra (in columns) with estimated concentrations. Columns are in the same order than `sample.name` and rows correspond to points of `ppm.grid`.

`quantification` Data-frame with identified metabolites and their relative concentrations.

`deformed.library` A data frame containing the deformed library of each sample.

Methods

Multiple methods can be applied to Spectra objects.

- As usual for S4 object, show and summary methods are available, see Object summary
- All slots have an accessor `get_slot name`, see Accessors
- Two objects can be combined or a subset can be extracted, see Combine and subset methods
- All spectra contained in an object can be represented in a plot, see Visualisation methods

See Also

Spectra

ASICSUsersGuide *View ASICS User's Guide*

Description

Open the ASICS User's Guide (with default browser)

Usage

```
ASICSUsersGuide(view = TRUE)
```

Arguments

view	Logical. If TRUE, the user's guide will be opened with system default browser.

Details

The function `vignette("ASICS")` will find the short ASICS vignette that describes the main functions and how to obtain the ASICS User's Guide.
The User's Guide is not itself a true vignette because it is not automatically generated during the package build process.
If the operating system is not Windows, then the HTML viewer used is the one given by `Sys.getenv("R_BROWSER")`.
The HTML viewer can be changed using `Sys.setenv(R_BROWSER = )`.

Value

Character string giving the file location. If `view = TRUE`, the HTML viewer is started and the User's Guide is opened, as a side effect.

Examples

```
# To get the location
ASICSUsersGuide(view = FALSE)

# To open in a HTML viewer
## Not run: ASICSUsersGuide()
```

binning *Binning/Bucketing of NMR spectra*

Description

Apply a binning function on a spectrum.

Usage

```
binning(
  spectra,
  bin = 0.01,
  exclusion.areas = matrix(c(4.5, 5.1), ncol = 2),
  normalisation = TRUE,
  low.lim = 0.5,
  high.lim = 10,
  ncores = 1,
  verbose = TRUE,
  ...
)
```

Arguments

spectra	Data frame with spectra in columns and chemical shifts in rows. Colnames of this data frame correspond to sample names and rownames to chemical shift grid (in ppm).
bin	Numeric value specifying the bin width.
exclusion.areas	Definition domain of spectra that have to be excluded of the analysis (ppm). By default, the water region is excluded (4.5-5.1 ppm).
normalisation	Logical. If TRUE a normalisation is applied for each spectrum (see normaliseSpectra for details). Default to TRUE.
low.lim, high.lim	low and high chemical shift limits for the output bins (default values : low.lim = 0.5 and high.lim = 10).
ncores	Number of cores used in parallel evaluation. Default to 1.
verbose	A boolean value to allow print out process information.
...	Further arguments to be passed to the function normaliseSpectra

Value

A data frame with normalised spectra in columns and buckets in rows (bucket names correspond to the center of the bucket).

Examples

```
current_path <- system.file("extdata", package = "ASICS")
spectra_data <- importSpectra(name.dir = current_path,
                     name.file = "spectra_example.txt", type.import = "txt")
spectra_bin <- binning(spectra_data, bin = 0.01, type.norm = "pqn")
```

combineAndSubset-methods *Combine or subset functions*

Description

Methods available to combine multiple objects or to extract a subset of one object in ASICS package.

Usage

```
## S4 method for signature 'Spectra,ANY,ANY,ANY'
x[i]

## S4 method for signature 'Spectra'
c(x, ...)

## S4 method for signature 'ASICSResults,ANY,ANY,ANY'
x[i]

## S4 method for signature 'ASICSResults'
c(x, ...)

## S4 method for signature 'PureLibrary,ANY,ANY,ANY'
x[i]

## S4 method for signature 'PureLibrary'
c(x, ...)
```

Arguments

x	An object of class Spectra, PureLibrary or ASICSResults.
i	vector of indices specifying which elements to extract
...	objects to be concatenated

Value

A Spectra object containing a part of the original object or combining other Spectra objects

Examples

```
# Import data and create object
current_path <- file.path(system.file("extdata", package = "ASICS"),
                          "spectra_example.txt")
spectra_data <- read.table(current_path, header = TRUE, row.names = 1)
spectra_obj <- createSpectra(spectra_data)
```

```
# Extract the first sample
spectra_obj[1]
```

`createPureLibrary` *Create a pure library*

Description

Create a new pure library from a data frame containing different spectra of pure metabolites. The noise is removed by thresholding each spectrum during the creation of a new pure library.

Usage

```
createPureLibrary(spectra, nb.protons, threshold = 1)
```

Arguments

`spectra`	Data frame with spectra in columns and chemical shifts in rows. Colnames of this data frame correspond to pure metabolite names and rownames to chemical shift grid (in ppm).
`nb.protons`	Numeric vector of the number of protons for each pure metabolite spectrum contained in `spectra` data frame.
`threshold`	Numeric value or numeric vector of length `ncol(spectra)` below which pure spectrum values are considered to be zero. Default to 1.

Value

A PureLibrary object with the newly created library.

Examples

```
pure_spectra <- importSpectraBruker(system.file("extdata",
                                                "example_library",
                                                package = "ASICS"))
new_pure_library <- createPureLibrary(pure_spectra,
                                      nb.protons = c(5, 4))
```

createSpectra *Create a Spectra object*

Description

Create a new spectra object used for quantification.

Usage

```
createSpectra(spectra, norm.method = NULL, norm.params = NULL)
```

Arguments

spectra	Data frame with spectra in columns and chemical shifts in rows. Colnames of this data frame correspond to pure metabolite names and rownames to chemical shift grid (in ppm).
norm.method	Character specifying the normalisation method to use on spectra ONLY if the importSpectra function was not used.
norm.params	List containing normalisation parameteres (see normaliseSpectra for details) ONLY if the importSpectra function was not used.

Value

A Spectra object with spectra to quantify.

See Also

Spectra

Examples

```
current_path <- system.file("extdata", "example_spectra", package = "ASICS")
spectra_data <- importSpectraBruker(current_path)
spectra_obj <- createSpectra(spectra_data)
```

formatForAnalysis *Format data for analysis*

Description

Create an object of class SummarizedExperiment to use in functions pca, oplsda or kruskalWallis.

Usage

```
formatForAnalysis(
  data,
  design = NULL,
  feature_info = NULL,
  zero.threshold = 100,
  zero.group = NULL,
  outliers = NULL
)
```

Arguments

data	A data frame containing omics dataset with samples in columns and features of interest in rows (metabolites/buckets...).
design	A data frame describing the colums of data with at least two columns, the first one corresponding to the column names of data. Default to NULL (in which case, the column names of data are used for study design).
feature_info	A data frame describing the rows of data with at least two columns, the first one corresponding to the row names of data. Default to NULL (in which case, the row names of data are used for feature information).
zero.threshold	Remove features having a proportion of zeros larger than or equal to zero.threshold. Default to 100.
zero.group	Variable name of design data frame specifying the group variable used to remove features with a proportion of zeros larger than or equal to zero.threshold within the group. Default to NULL, no group.
outliers	Names of the outliers (samples) to remove.

Value

An object of type SummarizedExperiment with metabolite data given as buckets or quantified metabolites.

Examples

```
# Import quantification results
if (require("ASICSdata", quietly = TRUE)) {
  quantif_path <- system.file("extdata", "results_ASICS.txt",
                              package = "ASICSdata")
  quantification <- read.table(quantif_path, header = TRUE, row.names = 1)

  # Import design
  design <- read.table(system.file("extdata", "design_diabete_example.txt",
                                   package = "ASICSdata"), header = TRUE)

  # Create object for analysis and remove features with more than 25% of
  # zeros
  analysis_obj <- formatForAnalysis(quantification,
                                    design = design,
```

```
                                    zero.threshold = 25,
                                    zero.group = "condition")
}
```

`importSpectra` *Import metabolomic spectra*

Description

Import spectra from text or CSV, fid or 1r (preprocessed spectrum) files. (optional) Spectra are baseline corrected, aligned and normalised during the importation.

Usage

```
importSpectra(
  name.dir = NULL,
  name.file = NULL,
  type.import,
  baseline.correction = FALSE,
  alignment = FALSE,
  normalisation = TRUE,
  ncores = 1,
  verbose = TRUE,
  ...
)
```

Arguments

`name.dir`	Path of the folder containing the spectra. Each subfolder contains the fid or the 1r (preprocessed spectrum) files of this sample if `type = "fid"` or `type = "1r"`.
`name.file`	Name of the txt or csv file containing the spectra in columns (spectrum names in the first line and ppm grid in the first column).
`type.import`	Type of import. Either `"txt"`, `"csv"`, `"fid"` or `"1r"`.
`baseline.correction`	Logical. If `TRUE` a baseline correction is applied for each spectrum. Default to `FALSE`.
`alignment`	Logical. If `TRUE` a peak alignment is applied between all spectra. Default to `FALSE`.
`normalisation`	Logical. If `TRUE` a normalisation is applied for each spectrum (see `normaliseSpectra` for details). Default to `TRUE`.
`ncores`	Number of cores used in parallel evaluation. Default to `1`.
`verbose`	A boolean value to allow print out process information.
`...`	Further arguments to be passed to the functions `read.table`, `importSpectraBruker`, `Normalization` (`PepsNMR-package`), `alignSpectra` or `normaliseSpectra` for specifying the parameters of the algorithm, if necessary.

Details

Some preprocessing steps are included during the importation. First, spectra are baseline corrected if `baseline.correction = TRUE`. Then, all spectrum definition domains are aligned to a unique one (either the one specified in `ppm.grid` or the grid of the default library). Finally, all spectra are normalised if `normalisation = TRUE` and aligned if `alignment = TRUE`.

Value

A data frame with spectra in columns and chemical shifts (in ppm) in rows.

References

Wang, K.C., Wang, S.Y., Kuo, C.H., Tseng Y.J. (2013). Distribution-based classification method for baseline correction of metabolomic 1D proton nuclear magnetic resonance spectra. *Analytical Chemistry*, **85**(2), 1231-1239.

See Also

`importSpectraBruker` `normaliseSpectra` `alignSpectra`

Examples

```
## Import from txt file
current_path <- system.file("extdata", package = "ASICS")
spectra_data <- importSpectra(name.dir = current_path,
                    name.file = "spectra_example.txt", type.import = "txt")

## Import from fid file
current_path <- system.file("extdata", "example_spectra", package = "ASICS")
spectra_data <- importSpectra(name.dir = current_path, type.import = "fid",
                              subdirs = TRUE, dirs.names = TRUE)

## Import from txt file
current_path <- system.file("extdata", "example_spectra", package = "ASICS")
spectra_data <- importSpectra(name.dir = current_path, type.import = "1r")
```

`importSpectraBruker` *Import preprocessed metabolomic spectra from Bruker files*

Description

Import preprocessed spectra from Bruker files contained in a single folder. This folder contains one subfolder for each sample. (optional) Spectra are baseline corrected, aligned and normalised by the area under the curve during the importation.

Usage

```
importSpectraBruker(
  name.dir,
  which.spectra = "first",
  ppm.grid = NULL,
  sample.names = NULL,
  ncores = 1,
  verbose = TRUE
)
```

Arguments

name.dir	Path of the folder containing one subfolder by sample. Each subfolder contains the Bruker files of this sample.
which.spectra	If there is no folder with experiment number (all_spectra/<spectrum_name>/pdata/...) set which.spectra to NULL. Else if there is more than one spectrum by sample (all_spectra/<spectrum_name>/<experiment_number>/pdata/...), which is the spectrum to import (either always the first one with which.spectra = "first", always the last one with which.spectra = "last" or a vector of length the number of spectra that specifies the number of each spectrum to import). Default to "first".
ppm.grid	Numeric vector of a unique grid (definition domain) for all spectra (in ppm). Default to NULL (in which case, the default grid of the pure library is used).
sample.names	Character vector of sample names. Default to NULL (in which case, folder names are used).
ncores	Number of cores used in parallel evaluation. Default to 1.
verbose	A boolean value to allow print out process information.

Value

A data frame with spectra in columns and chemical shifts (in ppm) in rows.

See Also

normaliseSpectra alignSpectra

Examples

```
current_path <- system.file("extdata", "example_spectra", package = "ASICS")
spectra_data <- importSpectraBruker(current_path)
```

kruskalWallis *Kruskal-Wallis rank sum tests on a* SummarizedExperiment *object*

Description

Perform Kruskal-Wallis tests on a SummarizedExperiment object obtained with the formatForAnalysis function

Usage

```
kruskalWallis(
  analysis_data,
  condition,
  alpha = 0.05,
  type.data = "quantifications",
  ...
)
```

Arguments

analysis_data	A SummarizedExperiment object obtained with the formatForAnalysis function.
condition	The name of the design variable (two level factor) specifying the group of each sample.
alpha	Cutoff for adjusted p-values. Default to 0.05.
type.data	Type of data used for the analyses (*e.g.*,
...	Arguments to be passed to p.adjust such as the correction method to use with the method argument. "quantifications", "buckets"...). Default to "quantifications".

Value

A S4 object of class AnalysisResults containing test results.

See Also

AnalysisResults

Examples

```
# Import quantification results
if (require("ASICSdata", quietly = TRUE)) {
  quantif_path <- system.file("extdata", "results_ASICS.txt",
                              package = "ASICSdata")
  quantification <- read.table(quantif_path, header = TRUE, row.names = 1)

  # Import design
```

```
  design <- read.table(system.file("extdata", "design_diabete_example.txt",
                                   package = "ASICSdata"), header = TRUE)
  design$condition <- factor(design$condition)

  # Create object for analysis and remove features with more than 25% of
  # zeros
  analysis_obj <- formatForAnalysis(quantification,
                                    zero.threshold = 25, design = design)
  res_tests <- kruskalWallis(analysis_obj, "condition", method = "BH")
}
```

`normaliseSpectra` *Normalisation*

Description

Normalise a data frame of spectra to a constant sum (CS) or with a method of `PepsNMR` package (see `Normalization`).

Usage

```
normaliseSpectra(spectra, type.norm = "CS", verbose = TRUE, ...)
```

Arguments

`spectra`	Data frame with spectra in columns and chemical shifts in rows. Colnames of this data frame correspond to pure metabolite names and rownames to chemical shift grid (in ppm).
`type.norm`	Type of normalisation : "CS", "mean", "pqn", "median", "firstquartile" or "peak". Default to "CS".
`verbose`	A boolean value to allow print out process information.
`...`	other arguments to be passed to `Normalization`

Value

A data frame with normalised spectra in columns and chemical shifts (in ppm) in rows.

Examples

```
current_path <- system.file("extdata", package = "ASICS")
spectra_data <- importSpectra(name.dir = current_path,
                     name.file = "spectra_example.txt", type.import = "txt")
spectra_norm <- normaliseSpectra(spectra_data, type.norm = "pqn")
```

`oplsda` *Orthogonal projections to latent structures discriminant analysis (OPLS-DA) on a* `SummarizedExperiment` *object*

Description

Perform an OPLS-DA with the function of the `ropls` package on a `SummarizedExperiment` object obtained with the `formatForAnalysis` function

Usage

```
oplsda(
  analysis_data,
  condition,
  cross.val = 1,
  thres.VIP = 1,
  type.data = "quantifications",
  seed = 12345,
  ...
)
```

Arguments

`analysis_data`	A `SummarizedExperiment` object obtained with the `formatForAnalysis` function.
`condition`	The name of the design variable (two level factor) specifying the response to be explained.
`cross.val`	Number of cross validation folds.
`thres.VIP`	A number specifying the VIP threshold used to identify influential variables.
`type.data`	Type of data used for the analyses (*e.g.,* `"quantifications"`, `"buckets"`...). Default to `"quantifications"`.
`seed`	Random seed to control randomness of cross validation folds.
`...`	Further arguments to be passed to the function `opls` for specifying the parameters of the algorithm, if necessary.

Value

A S4 object of class AnalysisResults containing OPLS-DA results.

References

Trygg, J. and Wold, S. (2002). Orthogonal projections to latent structures (O-PLS). *Journal of Chemometrics*, **16**(3), 119–128.

Thevenot, E.A., Roux, A., Xu, Y., Ezan, E., Junot, C. 2015. Analysis of the human adult urinary metabolome variations with age, body mass index and gender by implementing a comprehensive workflow for univariate and OPLS statistical analyses. *Journal of Proteome Research*. **14**:3322-3335.

See Also

AnalysisResults

Examples

```
# Import quantification results
if (require("ASICSdata", quietly = TRUE)) {
  quantif_path <- system.file("extdata", "results_ASICS.txt",
                              package = "ASICSdata")
  quantification <- read.table(quantif_path, header = TRUE, row.names = 1)

  # Import design
  design <- read.table(system.file("extdata", "design_diabete_example.txt",
                                   package = "ASICSdata"), header = TRUE)
  design$condition <- factor(design$condition)

  # Create object for analysis and remove features with more than 25% of
  # zeros
  analysis_obj <- formatForAnalysis(quantification,
                                    zero.threshold = 25, design = design)
  res_oplsda <- oplsda(analysis_obj, "condition", orthoI = 1)
}
```

---

pca *Principal Component Analysis (PCA) on a* SummarizedExperiment *object*

Description

Perform a PCA with the function of the ropls package on a SummarizedExperiment object obtained from the formatForAnalysis function

Usage

```
pca(
  analysis_data,
  scale.unit = TRUE,
  type.data = "quantifications",
  condition = NULL
)
```

Arguments

analysis_data	A SummarizedExperiment object obtained from the formatForAnalysis function.
scale.unit	Logical. If TRUE, data are scaled to unit variance prior PCA.

type.data	Type of data used for the analysis (*e.g.,* "quantifications", "buckets"...). Default to "quantifications".
condition	The name of the design variable (two level factor) specifying the groups, if one is available. Default to NULL, no group provided.

Value

A S4 object of class AnalysisResults containing PCA results.

See Also

AnalysisResults

Examples

```
# Import quantification results
if (require("ASICSdata", quietly = TRUE)) {
  quantif_path <- system.file("extdata", "results_ASICS.txt",
                              package = "ASICSdata")
  quantification <- read.table(quantif_path, header = TRUE, row.names = 1)

  # Create object for analysis and remove features with more than 25% of
  # zeros
  analysis_obj <- formatForAnalysis(quantification, zero.threshold = 25)
  res_pca <- pca(analysis_obj)
}
```

---

plotAlignment *Tile plot*

Description

Tile plot of spectra to see if an alignment is needed or the result of an alignment.

Usage

```
plotAlignment(spectra_obj, xlim = c(0, 10))
```

Arguments

spectra_obj	An object of class Spectra obtained with the function createSpectra.
xlim	Boundaries for x.

Value

A ggplot plot of original and reconstructed spectra of one sample in the same figure for ASICSResults object. In addition, one pure metabolite spectrum (as provided in the reference library) and the deformed one can be superimposed to the plot.

See Also

alignSpectra

Examples

```
# Import data and create object
current_path <- system.file("extdata", "example_spectra", package = "ASICS")
spectra_data <- importSpectraBruker(current_path)
spectra_obj <- createSpectra(spectra_data)
plotAlignment(spectra_obj, xlim = c(3,4))
```

PureLibrary-class *Class* PureLibrary

Description

Objects of class PureLibrary contain a set of pure metabolite NMR spectra, used as a reference for the quantification. This class is an extension of the class Spectra, with an additional slot (number of protons for each metabolite) needed for spectrum quantification.

Slots

nb.protons Numeric vector of the number of protons of each pure metabolite spectra.

Methods

Multiple methods can be applied on PureLibrary objects.

- As usual for S4 object, show and summary methods are available, see Object summary
- All slots have an accessor get_slot name, see Accessors
- Two objects can be combined or a subset can be extracted, see Combine and subset methods
- All spectra contained in an object can be represented in a plot, see Visualisation methods

See Also

Spectra

pure_library *Pure spectra library*

Description

The 1D 1H NMR spectra of 191 reference compounds were collected to build the default library of reference spectra. These compounds have been prepared and measured using a Bruker Avance III HD spectrometer in the MetaToul - AXIOM Site at Toulouse (France). For more details on the preparation, please see Tardivel et al. (2017).

Format

A PureLibrary object with 4 entries:

sample.name names of the metabolites

ppm.grid common grid for all spectra

spectra data frame with each pure metabolite spectrum in column

nb.protons number of protons of each metabolite

References

Tardivel P., Canlet C., Lefort G., Tremblay-Franco M., Debrauwer L., Concordet D., Servien R. (2017). ASICS: an automatic method for identification and quantification of metabolites in complex 1D 1H NMR spectra. *Metabolomics*, **13**(10): 109. https://doi.org/10.1007/s11306-017-1244-5

simulate_spectra *Simulate a set of spectra*

Description

Simulate a set of spectra based on the default library with shifts

Usage

```
simulate_spectra(
  n.spectra,
  max.shift = 0.02,
  metab.percent = 0.5,
  metab.different = 4,
  add.noise = 0.07,
  mult.noise = 0.09
)
```

Arguments

`n.spectra`	Number of spectra to simulate.
`max.shift`	Maximum shift allowed for artificial deformation of pure spectra (default to 0.02).
`metab.percent`	Percentage of present metabolites in complex spectra (default to 0.5).
`metab.different`	Number of metabolites that are different between each complex spectra (default to 4).
`add.noise`, `mult.noise`	additive and multiplicative noises. By default, `add.noise` = `0.15` and `mult.noise` = `0.172`

Value

A list with a data frame of simulated spectra in columns and a data frame of simulated quantifications.

Examples

```
spectra <- simulate_spectra(n.spectra = 10)
```

Spectra-class *Class Spectra*

Description

Objects of class Spectra contain a set of NMR spectra. It includes preprocessed spectra and can be created with the function `createSpectra`.

Slots

`sample.name` Character vector of sample names.

`ppm.grid` Numeric vector of a unique grid (definition domain) for all spectra (in ppm).

`spectra` Numeric matrix with all spectra in columns. Columns must be in the same order as for `sample.name` and rows correspond to points of `ppm.grid`.

`norm.method` Character specifying the normalisation method to use on spectra

`norm.params` List containing normalisation parameteres (see `normaliseSpectra` for details).

Methods

Multiple methods can be applied on Spectra objects.

- As usual for S4 object, show and summary methods are available, see Object summary
- All slots have an accessor `get_slot name`, see Accessors
- Two objects can be combined or a subset can be extracted, see Combine and subset methods
- All spectra contained in an object can be represented in a plot, see Visualisation methods

summary-methods *Summary methods*

Description

Methods available to summarize the various S4 objects of ASICS package.

Usage

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

## S4 method for signature 'Spectra'
summary(object)

## S4 method for signature 'Spectra'
length(x)

## S4 method for signature 'Spectra'
dim(x)

## S4 method for signature 'ASICSResults'
show(object)

## S4 method for signature 'ASICSResults'
dim(x)

## S4 method for signature 'AnalysisResults'
show(object)

## S4 method for signature 'AnalysisResults'
summary(object)
```

Arguments

object	An object of class Spectra, PureLibrary, ASICSResults or AnalysisResults.
x	An object of class Spectra, PureLibrary or ASICSResults.

Value

A summary of the object, its length or its dimensions.

Examples

```
# Import data and create object
current_path <- file.path(system.file("extdata", package = "ASICS"),
                          "spectra_example.txt")
```

```
spectra_data <- read.table(current_path, header = TRUE, row.names = 1)
spectra_obj <- createSpectra(spectra_data)

# Summary
summary(spectra_obj)
# Length
length(spectra_obj)
# Dimensions
dim(spectra_obj)
```

`visualisation-methods-analyses`
Visualisation methods

Description

Method available to plot results of analyses in ASICS package.

Usage

```
## S4 method for signature 'AnalysisResults,ANY'
plot(
  x,
  y,
  ...,
  graph = c("default", "ind", "var", "eig", "boxplot", "buckets"),
  add.label = TRUE,
  n.label.var = 10,
  axes = c(1, 2),
  col.ind = NULL,
  xlim = c(0.5, 10),
  ylim = NULL
)
```

Arguments

`x`	An object of class AnalysisResults.
`y`	Currently not used.
`...`	Currently not used.
`graph`	A vector specifying what to plot. Allowed values are `"eig"` for the scree-graph (PCA), `"ind"` for plot of individuals (PCA and OPLS-DA), `"var"` for plot of variables (PCA and OPLS-DA), `"boxplot"` for boxplots of test results and `"buckets"` to show significant or influential buckets on the mean spectrum. Default value is `"default"` (*i.e.,* `c("ind", "var")` for PCA and OPLS-DA and `c("boxplot")` for tests).
`add.label`	If `TRUE`, labels are added on individual plot.
`n.label.var`	An integer indicating the number of label to add on variable plot.

axes	A numeric vector of length 2 specifying the dimensions to be plotted for individual and variable plots.
col.ind	A character specifying the name of the design variable used to color the observations by groups for PCA individual plot.
xlim, ylim	Boundaries for x and y, respectively.

Value

- PCA: a ggplot plot that allows for the visualisation of PCA results (eigen values, individuals and variables)
- OPLS-DA: a ggplot plot that allows for the visualisation of OPLS-DA results (individuals and variables). If cross.val > 1 in oplsda, the best model is plotted.

Examples

```
# Import quantification results
if (require("ASICSdata", quietly = TRUE)) {
  quantif_path <- system.file("extdata", "results_ASICS.txt",
                              package = "ASICSdata")
  quantification <- read.table(quantif_path, header = TRUE, row.names = 1)

  # Import design
  design <- read.table(system.file("extdata", "design_diabete_example.txt",
                                   package = "ASICSdata"), header = TRUE)
  design$condition <- factor(design$condition)

  # Create object for analysis and remove metabolites with more than 25% of
  # zeros
  analysis_obj <- formatForAnalysis(quantification,
                                    zero.threshold = 25, design = design)

  # Perform a PCA and plot results
  res_pca <- pca(analysis_obj)
  plot(res_pca)

  # Perform an OPLS-DA and plot results
  res_oplsda <- oplsda(analysis_obj, "condition", orthoI = 1)
  plot(res_oplsda)
}
```

---

visualisation-methods-spectra

Visualisation methods

Description

Methods available to plot one object in ASICS package.

Usage

```
## S4 method for signature 'Spectra,ANY'
plot(x, y, xlim = c(0.5, 10), ylim = NULL, ...)

## S4 method for signature 'ASICSResults,ANY'
plot(
  x,
  y,
  idx = 1,
  xlim = c(0.5, 10),
  ylim = NULL,
  pure.library = NULL,
  add.metab = NULL,
  ...
)
```

Arguments

x	An object of class Spectra, PureLibrary or ASICSResults.
y	Currently not used.
xlim, ylim	Boundaries for x and y, respectively.
...	Currently not used.
idx	Index of the spectrum to plot. Default to 1.
pure.library	Pure library used for the quantification. Default to NULL (in which case, the library included in the package is used).
add.metab	Name of one metabolite to add to the plot. Default to NULL (in which case, no pure spectrum added to the plot).

Value

- A ggplot plot of all spectra (or of a subset) on the same figure for Spectra and PureLibrary objects.
- A ggplot plot of original and reconstructed spectra of one sample in the same figure for ASICSResults object. In addition, one pure metabolite spectrum (as provided in the reference library) and the deformed one can be superimposed to the plot.

Examples

```
# Import data and create object
current_path <- system.file("extdata", "example_spectra", package = "ASICS")
spectra_data <- importSpectraBruker(current_path)
spectra_obj <- createSpectra(spectra_data)
spectra_obj <- createSpectra(spectra_data)

# Plot the spectra
plot(spectra_obj)
```

Index