

# Package ‘ASICS’

October 15, 2018

**Type** Package

**Title** Automatic Statistical Identification in Complex Spectra

**Version** 1.0.1

**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, grDevices, gridExtra, methods, plyr, quadprog, ropls, speaq, stats, SummarizedExperiment, utils, zoo

**License** GPL (>= 2)

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 6.0.1

**Suggests** knitr, rmarkdown, BiocStyle, testthat, ASICSdata

**VignetteBuilder** knitr

**Collate** 'ASICS.R' 'Spectra-class.R' 'ASICSResults-class.R'  
'AnalysisResults-class.R' 'PureLibrary-class.R'  
'analysis\_on\_quantification.R' 'baseline\_correction.R'  
'concentration\_optimisation.R' 'data.R'  
'library\_transformation.R' 'load\_data.R' 'plot\_spectrum.R'  
'user\_guide.R' 'utils.R'

**biocViews** Software, DataImport, Cheminformatics, Metabolomics

**git\_url** <https://git.bioconductor.org/packages/ASICS>

**git\_branch** RELEASE\_3\_7

**git\_last\_commit** 4a113d6

**git\_last\_commit\_date** 2018-08-21

**Date/Publication** 2018-10-15

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

**Maintainer** Gaëlle Lefort <gaelle.lefort@inra.fr>

**R topics documented:**

accessors-methods . . . . .	2
alignment . . . . .	3
AnalysisResults-class . . . . .	4
ASICS . . . . .	5
ASICSResults-class . . . . .	6
ASICSUsersGuide . . . . .	7
baselineCorrection . . . . .	8
binning . . . . .	8
combineAndSubset-methods . . . . .	9
createPureLibrary . . . . .	10
createSpectra . . . . .	11
formatForAnalysis . . . . .	11
importSpectraBruker . . . . .	12
kruskalWallis . . . . .	14
normalisation . . . . .	15
oplsda . . . . .	15
pca . . . . .	17
PureLibrary-class . . . . .	18
pure_library . . . . .	18
Spectra-class . . . . .	19
summary-methods . . . . .	19
visualisation-methods-analyses . . . . .	20
visualisation-methods-spectra . . . . .	22

<b>Index</b>	<b>23</b>
--------------	-----------

---

accessors-methods	<i>Accessors</i>
-------------------	------------------

---

**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 '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)
```

## Description

Align spectra of a data frame with the CluPA algorithm (Vu et al., (2011))

## Usage

```
alignment(spectra, baseline.threshold = 0.02, reference = NULL,  
max.shift = 0.002)
```

## 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).
baseline.threshold	Value of baseline threshold used to identify peaks. Default to 0.02.
reference	Index of the reference spectrum used for the alignment. Default to NULL, <i>i.e.</i> the reference spectrum is automatically detected with the <a href="#">findRef</a> function of the <a href="#">speaq</a> package.
max.shift	Maximum shift allowed for the alignment. Default to 0.002.

## 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., Dommissie, 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 <- file.path(system.file("extdata", package = "ASICS"),  
"spectra_example.txt")  
spectra_data <- read.table(current_path, header = TRUE, row.names = 1)  
spectra_base_cor <- alignment(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, threshold.noise = 0.02,
      seed = 1234, ncores = 1)
```

**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.
- `threshold.noise` Threshold for signal noise. Default to 0.02.
- `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.

**Value**

An object of type [ASICSResults](#) containing the quantification results.

**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 <- 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)

# Estimation of relative quantification of Lactate and L-Alanine
to_exclude <- matrix(c(4.5, 10), ncol = 2)
pure_lib <- pure_library[getSampleName(pure_library) %in%
                        c("Lactate", "L-Alanine")]
resASICS <- ASICS(spectra_obj[1], exclusion.areas = to_exclude,
                  pure.library = pure_lib)
```

---

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()
```

---

baselineCorrection	<i>Baseline correction</i>
--------------------	----------------------------

---

**Description**

Apply a baseline correction to a spectra with the algorithm described in Wang et al. (2013).

**Usage**

```
baselineCorrection(spectra, ncores = 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).
ncores	Number of cores used in parallel evaluation. Default to 1.

**Value**

A data frame with baseline corrected 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.

**Examples**

```
current_path <- file.path(system.file("extdata", package = "ASICS"),
                           "spectra_example.txt")
spectra_data <- read.table(current_path, header = TRUE, row.names = 1)
spectra_base_cor <- baselineCorrection(spectra_data)
```

---

binning	<i>Binning/Bucketing of NMR spectra</i>
---------	---

---

**Description**

Apply a binning function on a spectrum.

**Usage**

```
binning(spectra, bin = 0.01, exclusion.areas = matrix(c(4.5, 5.1), ncol =
2), ncores = 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).
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).
ncores	Number of cores used in parallel evaluation. Default to 1.

**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 <- file.path(system.file("extdata", package = "ASICS"),
                           "spectra_example.txt")
spectra_data <- read.table(current_path, header = TRUE, row.names = 1)
spectra_bin <- binning(spectra_data, bin = 0.01)
```

---

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 <a href="#">Spectra</a> , <a href="#">PureLibrary</a> or <a href="#">ASICSResults</a> .
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 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)
```

**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).

**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**

<code>data</code>	A data frame containing omics dataset with samples in columns and features of interest in rows (metabolites/buckets...).
<code>design</code>	A data frame describing the columns 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).
<code>feature_info</code>	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).
<code>zero.threshold</code>	Remove features having a proportion of zeros larger than or equal to <code>zero.threshold</code> . Default to 100.
<code>zero.group</code>	Variable name of design data frame specifying the group variable used to remove features with a proportion of zeros larger than or equal to <code>zero.threshold</code> within the group. Default to NULL, no group.
<code>outliers</code>	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")
}
```

---

importSpectraBruker     *Import metabolomic spectra from Bruker files*

---

**Description**

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

## Usage

```
importSpectraBruker(name.dir, which.spectra = "last",
  baseline.correction = TRUE, alignment = FALSE, ppm.grid = NULL,
  sample.names = NULL, ncores = 1, ...)
```

## Arguments

<code>name.dir</code>	Path of the folder containing one subfolder by sample. Each subfolder contains the Bruker files of this sample.
<code>which.spectra</code>	If there is more than one spectrum by sample, which is the spectrum to import (either always the first one with <code>which.spectra = "first"</code> , always the last one with <code>which.spectra = "last"</code> or a vector of length the number of spectra that specifies the number of each spectrum to import). Default to "last".
<code>baseline.correction</code>	Logical. If TRUE a baseline correction is applied for each spectrum (Wang et al (2013)). Default to TRUE.
<code>alignment</code>	Logical. If TRUE a peak alignment is applied between all spectra (Vu et al (2011)). Default to FALSE.
<code>ppm.grid</code>	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).
<code>sample.names</code>	Character vector of sample names. Default to NULL (in which case, folder names are used).
<code>ncores</code>	Number of cores used in parallel evaluation. Default to 1.
<code>...</code>	Further arguments to be passed to the function <code>alignment</code> 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 by the area under the curve.

## 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.

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

## See Also

[baselineCorrection](#) [normalisation](#) [alignment](#)

**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 <a href="#">SummarizedExperiment</a> object obtained with the <a href="#">formatForAnalysis</a> 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 ( <i>e.g.</i> ,
...	Arguments to be passed to <a href="#">p.adjust</a> 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)

  # 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")
}

```

---

normalisation

*Normalisation*


---

### Description

Normalise a data frame of spectra by the area under the curve.

### Usage

```
normalisation(spectra)
```

### 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).

### Value

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

### Examples

```

current_path <- file.path(system.file("extdata", package = "ASICS"),
                          "spectra_example.txt")
spectra_data <- read.table(current_path, header = TRUE, row.names = 1)
spectra_norm <- normalisation(spectra_data)

```

---

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 <a href="#">SummarizedExperiment</a> object obtained with the <a href="#">formatForAnalysis</a> 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 ( <i>e.g.</i> , "quantifications", "buckets"...). Default to "quantifications".
seed	Random seed to control randomness of cross validation folds.
...	Further arguments to be passed to the function <a href="#">opls</a> 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)

  # 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	<i>Principal Component Analysis (PCA) on a <a href="#">SummarizedExperiment</a> object</i>
-----	--

---

### Description

Perform a PCA with the function of the [ropIs](#) 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 <a href="#">SummarizedExperiment</a> object obtained from the <a href="#">formatForAnalysis</a> 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)  
}
```

---

PureLibrary-class	<i>Class PureLibrary</i>
-------------------	--------------------------

---

### 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	<i>Pure spectra library</i>
--------------	-----------------------------

---

### Description

The 1D <sup>1</sup>H 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 <sup>1</sup>H NMR spectra. *Metabolomics*, **13**(10): 109. <https://doi.org/10.1007/s11306-017-1244-5>

---

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`.

### 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, axes = c(1, 2),  
     col.ind = NULL, xlim = c(0.5, 10), ylim = NULL)
```

**Arguments**

x	An object of class <a href="#">AnalysisResults</a> .
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>i.e.</i> , <code>c("ind", "var")</code> ) for PCA and OPLS-DA and <code>c("boxplot")</code> for tests).
add.label	If TRUE, labels are added on individual 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)

  # 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 <a href="#">Spectra</a> , <a href="#">PureLibrary</a> or <a href="#">ASICSResults</a> .
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

- [, ASICSResults, ANY, ANY, ANY-method  
(combineAndSubset-methods), 9
- [, PureLibrary, ANY, ANY, ANY-method  
(combineAndSubset-methods), 9
- [, Spectra, ANY, ANY, ANY-method  
(combineAndSubset-methods), 9
- [. ASICSResults  
(combineAndSubset-methods), 9
- [. PureLibrary  
(combineAndSubset-methods), 9
- [. Spectra (combineAndSubset-methods), 9
  
- Accessors, 5, 7, 18, 19
- accessors-methods, 2
- alignment, 3, 13
- AnalysisResults, 3–5, 14, 16, 17, 20, 21
- AnalysisResults-class, 4
- ASICS, 5
- ASICSResults, 3, 6, 10, 20, 22
- ASICSResults-class, 6
- ASICSUsersGuide, 7
  
- baselineCorrection, 8, 13
- binning, 8
  
- c, ASICSResults-method  
(combineAndSubset-methods), 9
- c, PureLibrary-method  
(combineAndSubset-methods), 9
- c, Spectra-method  
(combineAndSubset-methods), 9
- c. ASICSResults  
(combineAndSubset-methods), 9
- c. PureLibrary  
(combineAndSubset-methods), 9
- c. Spectra (combineAndSubset-methods), 9
- Combine and subset methods, 7, 18, 19
- combineAndSubset-methods, 9
- createPureLibrary, 10
- createSpectra, 5, 6, 11, 19
  
- dim, ASICSResults-method  
(summary-methods), 19
- dim, Spectra-method (summary-methods), 19
  
- dim.Spectra (summary-methods), 19
  
- findRef, 4
- formatForAnalysis, 11, 14–17
  
- getBestModel (accessors-methods), 2
- getBestModel, AnalysisResults-method  
(accessors-methods), 2
- getCVError (accessors-methods), 2
- getCVError, AnalysisResults-method  
(accessors-methods), 2
- getDataset (accessors-methods), 2
- getDataset, AnalysisResults-method  
(accessors-methods), 2
- getDeformedLibrary (accessors-methods),  
2
- getDeformedLibrary, ASICSResults-method  
(accessors-methods), 2
- getMeanByGroup (accessors-methods), 2
- getMeanByGroup, AnalysisResults-method  
(accessors-methods), 2
- getNbProtons (accessors-methods), 2
- getNbProtons, PureLibrary-method  
(accessors-methods), 2
- getPpmGrid (accessors-methods), 2
- getPpmGrid, Spectra-method  
(accessors-methods), 2
- getQuantification (accessors-methods), 2
- getQuantification, ASICSResults-method  
(accessors-methods), 2
- getReconstructedSpectra  
(accessors-methods), 2
- getReconstructedSpectra, ASICSResults-method  
(accessors-methods), 2
- getResults (accessors-methods), 2
- getResults, AnalysisResults-method  
(accessors-methods), 2
- getSampleName (accessors-methods), 2
- getSampleName, Spectra-method  
(accessors-methods), 2
- getSpectra (accessors-methods), 2
- getSpectra, Spectra-method  
(accessors-methods), 2
- getTypeAnalysis (accessors-methods), 2

- getTypeAnalysis, AnalysisResults-method
  - (accessors-methods), 2
- getTypeData (accessors-methods), 2
- getTypeData, AnalysisResults-method
  - (accessors-methods), 2
- ggplot, 21, 22
- importSpectraBruker, 12
- kruskalWallis, 4, 11, 14
- length, Spectra-method
  - (summary-methods), 19
- length.Spectra (summary-methods), 19
- normalisation, 13, 15
- Object summary, 5, 7, 18, 19
- opls, 5, 16
- oplsda, 4, 11, 15, 21
  
- p.adjust, 14
- pca, 4, 11, 17
- plot, AnalysisResults, ANY-method
  - (visualisation-methods-analyses), 20
- plot, ASICSResults, ANY-method
  - (visualisation-methods-spectra), 22
- plot, Spectra, ANY-method
  - (visualisation-methods-spectra), 22
- plot.AnalysisResults
  - (visualisation-methods-analyses), 20
- plot.ASICSResults
  - (visualisation-methods-spectra), 22
- plot.Spectra
  - (visualisation-methods-spectra), 22
- pure\_library, 6, 18
- PureLibrary, 3, 5, 10, 18, 20, 22
- PureLibrary-class, 18
  
- ropls, 5, 15, 17
  
- show, AnalysisResults-method
  - (summary-methods), 19
- show, ASICSResults-method
  - (summary-methods), 19
- show, Spectra-method (summary-methods), 19
  
- show.AnalysisResults (summary-methods), 19
- show.ASICSResults (summary-methods), 19
- show.Spectra (summary-methods), 19
- Spectra, 3, 5–7, 10, 11, 18–20, 22
- Spectra-class, 19
- SummarizedExperiment, 5, 11, 12, 14–17
- summary, AnalysisResults-method
  - (summary-methods), 19
- summary, Spectra-method
  - (summary-methods), 19
- summary-methods, 19
- summary.AnalysisResults
  - (summary-methods), 19
- summary.Spectra (summary-methods), 19
  
- Visualisation methods, 5, 7, 18, 19
- visualisation-methods-analyses, 20
- visualisation-methods-spectra, 22