Package: metabolyseR (via r-universe)

May 17, 2026

Title Methods for Pre-Treatment, Data Mining and Correlation Analyses of Metabolomics Data

Version 0.15.4

Description A tool kit for pre-treatment, modelling, feature selection and correlation analyses of metabolomics data.

URL https://jasenfinch.github.io/metabolyseR/

BugReports https://github.com/jasenfinch/metabolyseR/issues

biocViews Metabolomics

Depends R (>= 3.4.0)

Imports Hmisc, dplyr, missForest, randomForest, stringr, tibble, tidyr, methods, ggplot2, ggthemes, ggdendro, purrr, doFuture, e1071, forestControl, crayon, yaml, cli, lubridate, ggrepel, patchwork, yardstick, broom, rlang, tidyselect, magrittr, furrr, future

License GPL (>= 3)

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

Suggests knitr, rmarkdown, readr, testthat, metaboData, prettydoc, covr, kableExtra

Collate all-classes.R analysis-accessors.R aggregate.R analysisData.R correction.R correlations.R impute.R info.R join.R keep.R mds.R metabolyse.R metabolyseR.R modelling.R modelling-plots.R nlda.R occupancy.R parameters.R plotExplanatoryHeatmap.R plotFeature.R plotLDA.R plotOccupancy.R plotPCA.R plotRSD.R plotSupervisedRF.R plotTIC.R plotting.R plotUnsupervisedRF.R pre-treatment.R predict.R QC.R reexports.R remove.R randomForest-permute.R randomForest-classification.R randomForest-regression.R randomForest-internals.R randomForest-unsupervised.R randomForest.R rsd.R roc.R show-method.R split.R transform.R tune.R univariate.R modelling-accessors.R

VignetteBuilder knitr

Remotes jasenfinch/missForest

Config/pak/sysreqs cmake make libicu-dev libuv1-dev

Repository https://aberhrml.r-universe.dev

Date/Publication 2023-09-12 15:01:48 UTC

RemoteUrl https://github.com/aberHRML/metabolyseR

RemoteRef HEAD

RemoteSha 08bb28b7d2bb733e675a0a9fd88509b4b8241398

Contents

`aggregateMean` *Sample aggregation*

Description

Aggregation of sample features based on a grouping variable.

Usage

```
aggregateMean(d, cls = "class")

## S4 method for signature 'AnalysisData'
aggregateMean(d, cls = "class")

aggregateMedian(d, cls = "class")

## S4 method for signature 'AnalysisData'
aggregateMedian(d, cls = "class")

aggregateSum(d, cls = "class")

## S4 method for signature 'AnalysisData'
aggregateSum(d, cls = "class")
```

Arguments

`d`	S4 object of class `AnalysisData`
`cls`	info columns across which to aggregate the data

Details

Sample aggregation allows the electronic pooling of sample features based on a grouping variable. This is useful in situations such as the presence of technical replicates that can be aggregated to reduce the effects of pseudo replication.

Value

An S4 object of class `AnalysisData` containing the aggregated data.

Methods

- `aggregateMean`: Aggregate sample features to the group mean.
- `aggregateMedian`: Aggregate sample features to the group median.
- `aggregateSum`: Aggregate sample features to the group total.

Examples

```
## Each of the following examples shows the application of the aggregation method and then
## a Principle Component Analysis is plotted to show it's effect on the data structure.

## Initial example data preparation
library(metaboData)

d <- analysisData(abr1$neg[,200:300],abr1$fact) %>%
 occupancyMaximum(occupancy = 2/3)

d %>%
 plotPCA(cls = 'day')

## Mean aggregation
d %>%
 aggregateMean(cls = c('day','class')) %>%
 plotPCA(cls = 'day',ellipses = FALSE)

## Median aggregation
d %>%
 aggregateMedian(cls = c('day','class')) %>%
 plotPCA(cls = 'day',ellipses = FALSE)

## Sum aggregation
d %>%
 aggregateSum(cls = c('day','class')) %>%
 plotPCA(cls = 'day',ellipses = FALSE)
```

Analysis-class *Analysis S4 class*

Description

An S4 class to store analysis results.

Slots

`log` list containing analysis dates and time

`parameters` class AnalysisParameters containing the analysis parameters

`raw` list containing info and raw data

`pre-treated` list containing preTreated info and raw data

`modelling` list containing modelling results

`correlations` tibble containing weighted edgelist of correlations

analysisData *AnalysisData class constructor*

Description

Create an AnalysisData S4 object.

Usage

```
analysisData(data, info)
```

Arguments

`data`	table containing sample metabolomic data
`info`	table containing sample meta information

Value

An S4 object of class Analysis.

Examples

```
library(metaboData)
d <- analysisData(data = abr1$neg,info = abr1$fact)

print(d)
```

AnalysisData-class *AnalysisData S4 class*

Description

An S4 class for metabolomic data and sample meta information.

Slots

`data` sample metabolomic data

`info` sample meta information

analysisElements *Analysis elements*

Description

Return the analysis elements available in `metabolyseR`.

Usage

```
analysisElements()
```

Value

A character vector of analysis elements.

Examples

```
analysisElements()
```

analysisParameters *Create an* `AnalysisParameters` *S4 class object*

Description

Initiate an `AnalysisParameters` object with the default analysis parameters for each of the analysis elements.

Usage

```
analysisParameters(elements = analysisElements())
```

Arguments

elements	character vector containing elements for analysis.

Value

An S4 object of class AnalysisParameters containing the default analysis parameters.

Examples

```
p <- analysisParameters()

print(p)
```

AnalysisParameters-class *AnalysisParameters S4 class*

Description

An S4 class to store analysis parameters.

Slots

pre-treatment	list containing parameters for data pre-treatment
modelling	list containing parameters for modelling
correlations	list containing parameters for correlations

---

anova *ANOVA*

Description

One-way analysis of variance (ANOVA).

Usage

```
anova(
  x,
  cls = "class",
  pAdjust = "bonferroni",
  comparisons = list(),
  returnModels = FALSE
)

## S4 method for signature 'AnalysisData'
```

```
anova(
  x,
  cls = "class",
  pAdjust = "bonferroni",
  comparisons = list(),
  returnModels = FALSE
)
```

Arguments

x	S4 object of class AnalysisData
cls	a vector of sample info column names to analyse
pAdjust	p value adjustment method
comparisons	list of comparisons to perform
returnModels	should models be returned

Examples

```
library(metaboData)

d <- analysisData(abr1$neg[,200:300],abr1$fact)

## Perform ANOVA
anova_analysis <- anova(d,cls = 'day')

## Extract significant features
explanatoryFeatures(anova_analysis)
```

---

binaryComparisons *Modelling accessor methods*

Description

Methods for accessing modelling results.

Usage

```
binaryComparisons(x, cls = "class")

## S4 method for signature 'AnalysisData'
binaryComparisons(x, cls = "class")

mtry(x, cls = "class")

## S4 method for signature 'AnalysisData'
mtry(x, cls = "class")
```

```
type(x)

## S4 method for signature 'RandomForest'
type(x)

## S4 method for signature 'Univariate'
type(x)

response(x)

## S4 method for signature 'RandomForest'
response(x)

## S4 method for signature 'Univariate'
response(x)

metrics(x)

## S4 method for signature 'RandomForest'
metrics(x)

## S4 method for signature 'list'
metrics(x)

## S4 method for signature 'Analysis'
metrics(x)

predictions(x)

## S4 method for signature 'RandomForest'
predictions(x)

## S4 method for signature 'list'
predictions(x)

## S4 method for signature 'Analysis'
predictions(x)

importanceMetrics(x)

## S4 method for signature 'RandomForest'
importanceMetrics(x)

importance(x)

## S4 method for signature 'RandomForest'
importance(x)
```

```
## S4 method for signature 'Univariate'
importance(x)

## S4 method for signature 'list'
importance(x)

## S4 method for signature 'Analysis'
importance(x)

proximity(x, idx = NULL)

## S4 method for signature 'RandomForest'
proximity(x, idx = NULL)

## S4 method for signature 'list'
proximity(x, idx = NULL)

## S4 method for signature 'Analysis'
proximity(x, idx = NULL)

explanatoryFeatures(x, ...)

## S4 method for signature 'Univariate'
explanatoryFeatures(
  x,
  threshold = 0.05,
  value = c("adjusted.p.value", "p.value")
)

## S4 method for signature 'RandomForest'
explanatoryFeatures(
  x,
  metric = "false_positive_rate",
  value = c("value", "p-value", "adjusted_p-value"),
  threshold = 0.05
)

## S4 method for signature 'list'
explanatoryFeatures(x, ...)

## S4 method for signature 'Analysis'
explanatoryFeatures(x, ...)
```

Arguments

x	S4 object of class AnalysisData,RandomForest, Univariate, Analysis or a list.
cls	sample information column to use

idx	sample information column to use for sample names. If NULL, the sample row number will be used. Sample names should be unique for each row of data.
...	arguments to parse to method for specific class
threshold	threshold below which explanatory features are extracted
value	the importance value to threshold. See the usage section for possible values for each class.
metric	importance metric for which to retrieve explanatory features

Methods

- binaryComparisons: Return a vector of all possible binary comparisons for a given sample information column.
- mtry: Return the default mtry random forest parameter value for a given sample information column.
- type: Return the type of random forest analysis.
- response: Return the response variable name used for a random forest analysis.
- metrics: Retrieve the model performance metrics for a random forest analysis
- predictions: Retrieve the out of bag model response predictions for a random forest analysis.
- importanceMetrics: Retrieve the available feature importance metrics for a random forest analysis.
- importance: Retrieve feature importance results.
- proximity: Retrieve the random forest sample proximities.
- explanatoryFeatures: Retrieve explanatory features.

Examples

```
library(metaboData)

d <- analysisData(abr1$neg[,200:300],abr1$fact)

## Return possible binary comparisons for the `day` response column
binaryComparisons(d,cls = 'day')

## Return the default random forest `mtry` parameter for the `day` response column
mtry(d,cls = 'day')

## Perform random forest analysis
rf_analysis <- randomForest(d,cls = 'day')

## Return the type of random forest
type(rf_analysis)

## Return the response variable name used
response(rf_analysis)

## Retrieve the model performance metrics
metrics(rf_analysis)
```

```
## Retrieve the out of bag model response predictions
predictions(rf_analysis)

## Show the available feature importance metrics
importanceMetrics(rf_analysis)

## Retrieve the feature importance results
importance(rf_analysis)

## Retrieve the sample proximities
proximity(rf_analysis)

## Retrieve the explanatory features
explanatoryFeatures(rf_analysis,metric = 'false_positive_rate',threshold = 0.05)
```

---

bindRows *Bind* AnalysisData *objects by row*

Description

Bind the rows of AnalysisData objects contained within a list.

Usage

```
bindRows(d)

## S4 method for signature 'list'
bindRows(d)
```

Arguments

d	list object containing S4 objects of class AnalysisData to be bound

Value

An S4 object of class AnalysisData containg the bound data sets.

Examples

```
library(metaboData)
d <- list(
 negative = analysisData(abr1$neg,abr1$fact),
 positive = analysisData(abr1$pos,abr1$fact)
 )

bindRows(d)
```

changeParameter<- *Change analysis parameters*

Description

Change analysis parameters.

Usage

```
changeParameter(x, parameterName, elements = analysisElements()) <- value

## S4 replacement method for signature 'AnalysisParameters'
changeParameter(x, parameterName, elements = analysisElements()) <- value
```

Arguments

x	S4 object of class AnalysisParameters
parameterName	name of the parameter to change
elements	character vector of analysis elements to target parameter change. Can be any returned by analysisElements().
value	New value of the parameter

Details

For the parameter name selected, all parameters with that name will be altered.

Value

An S4 object of class AnalysisParameters.

Examples

```
p <- analysisParameters('pre-treatment')

changeParameter(p,'cls') <- 'day'

print(p)
```

clsAdd *Sample meta information wrangling*

Description

Query or alter sample meta information in `AnalysisData` or `Analysis` class objects.

Replace a given sample info column from an Analysis or AnalysisData object.

Usage

```
clsAdd(d, cls, value, ...)

## S4 method for signature 'AnalysisData'
clsAdd(d, cls, value)

## S4 method for signature 'Analysis'
clsAdd(d, cls, value, type = c("pre-treated", "raw"))

clsArrange(d, cls = "class", descending = FALSE, ...)

## S4 method for signature 'AnalysisData'
clsArrange(d, cls = "class", descending = FALSE)

## S4 method for signature 'Analysis'
clsArrange(
  d,
  cls = "class",
  descending = FALSE,
  type = c("pre-treated", "raw")
)

clsAvailable(d, ...)

## S4 method for signature 'AnalysisData'
clsAvailable(d)

## S4 method for signature 'Analysis'
clsAvailable(d, type = c("pre-treated", "raw"))

clsExtract(d, cls = "class", ...)

## S4 method for signature 'AnalysisData'
clsExtract(d, cls = "class")

## S4 method for signature 'Analysis'
clsExtract(d, cls = "class", type = c("pre-treated", "raw"))
```

```
clsRemove(d, cls, ...)

## S4 method for signature 'AnalysisData'
clsRemove(d, cls)

## S4 method for signature 'Analysis'
clsRemove(d, cls, type = c("pre-treated", "raw"))

clsRename(d, cls, newName, ...)

## S4 method for signature 'AnalysisData'
clsRename(d, cls, newName)

## S4 method for signature 'Analysis'
clsRename(d, cls, newName, type = c("pre-treated", "raw"))

clsReplace(d, value, cls = "class", ...)

## S4 method for signature 'AnalysisData'
clsReplace(d, value, cls = "class")

## S4 method for signature 'Analysis'
clsReplace(d, value, cls = "class", type = c("pre-treated", "raw"))
```

Arguments

d	S4 object of class Analysis or AnalysisData
cls	sample info column to extract
value	vactor of new sample information for replacement
...	arguments to pass to specific method
type	raw or pre-treated sample information
descending	TRUE/FALSE, arrange samples in descending order
newName	new column name

Methods

- clsAdd: Add a sample information column.
- clsArrange: Arrange sample row order by a specified sample information column.
- clsAvailable: Retrieve the names of the available sample information columns.
- clsExtract: Extract the values of a specified sample information column.
- clsRemove: Remove a sample information column.
- clsRename: Rename a sample information column.
- clsReplace: Replace a sample information column.

Examples

```
library(metaboData)
d <- analysisData(abr1$neg,abr1$fact)

## Add a sample information column named 'new'
d <- clsAdd(d,'new',1:nSamples(d))

print(d)

## Arrange the row orders by the 'day' column
d <- clsArrange(d,'day')

clsExtract(d,'day')

## Retreive the available sample information column names
clsAvailable(d)

## Extract the values of the 'day' column
clsExtract(d,'day')

## Remove the 'class' column
d <- clsRemove(d,'class')

clsAvailable(d)

## Rename the 'day' column to 'treatment'
d <- clsRename(d,'day','treatment')

clsAvailable(d)

## Replace the values of the 'treatment' column
d <- clsReplace(d,rep(1,nSamples(d)),'treatment')

clsExtract(d,'treatment')
```

---

correctionCenter *Batch/block correction*

Description

Correction of batch/block differences.

Usage

```
correctionCenter(d, block = "block", type = c("mean", "median"))

## S4 method for signature 'AnalysisData'
correctionCenter(d, block = "block", type = c("mean", "median"))
```

Arguments

d	S4 object of class AnalysisData
block	sample information column name to use containing sample block groupings
type	type of average to use

Details

There can sometimes be artificial batch related variability introduced into metabolomics analyses as a result of analytical instrumentation or sample preparation. With an appropriate randomised block design of sample injection order, batch related variability can be corrected using an average centring correction method of the individual features.

Value

An S4 object of class AnalysisData containing the corrected data.

Methods

- correctionCenter: Correction using group average centring.

Examples

```
## Initial example data preparation
library(metaboData)

d <- analysisData(abr1$neg[,200:300],abr1$fact) %>%
 occupancyMaximum(occupancy = 2/3)

## Group total ion count distributions prior to correction
d %>%
 plotTIC(by = 'day',colour = 'day')

## Group total ion count distributions after group median correction
d %>%
 correctionCenter(block = 'day',type = 'median') %>%
 plotTIC(by = 'day',colour = 'day')
```

---

correlations *Feature correlation analysis*

Description

Feature correlation analysis.

Usage

```
correlations(d, ...)

## S4 method for signature 'AnalysisData'
correlations(
  d,
  method = "pearson",
  pAdjustMethod = "bonferroni",
  corPvalue = 0.05,
  minCoef = 0,
  maxCor = Inf
)

## S4 method for signature 'Analysis'
correlations(d)
```

Arguments

d	S4 object of class AnalysisData
...	arguments to pass to specific method
method	correlation method. One of pearson or spearman.
pAdjustMethod	p-value adjustment method. See ?p.adjust for available methods.
corPvalue	p-value cut-off threshold for significance
minCoef	minimum absolute correlation coefficient threshold
maxCor	maximum number of returned correlations

Details

Correlation analyses can be used to identify associated features within data sets. This can be useful to identifying clusters of related features that can be used to annotate metabolites within data sets. All features are compared and the returned table of correlations are thresholded to the specified p-value cut-off.

Value

A tibble containing results of significantly correlated features.

Examples

```
library(metaboData)

d <- analysisData(abr1$neg[,200:300],abr1$fact)

correlations(d)
```

correlationsParameters *Correlations parameters*

Description

Retrieve the default parameters for correlation analysis.

Usage

```
correlationsParameters()
```

Examples

```
## Retrieve the default correlation parameters
p <- correlationsParameters()

## Assign the correlation parameters to analysis parameters
cp <- analysisParameters('correlations')
parameters(cp,'correlations') <- p

print(cp)
```

dat AnalysisData *and* Analysis *class accessors*

Description

Accessor methods for the AnalysisData and Analysis S4 classes.

Usage

```
dat(x, ...)

## S4 method for signature 'AnalysisData'
dat(x)

## S4 method for signature 'Analysis'
dat(x, type = c("pre-treated", "raw"))

dat(x, ...) <- value

## S4 replacement method for signature 'AnalysisData'
dat(x) <- value

## S4 replacement method for signature 'Analysis'
```

```
dat(x, type = c("pre-treated", "raw")) <- value

sinfo(x, ...)

## S4 method for signature 'AnalysisData'
sinfo(x)

## S4 method for signature 'Analysis'
sinfo(x, type = c("pre-treated", "raw"), value)

sinfo(x, ...) <- value

## S4 replacement method for signature 'AnalysisData'
sinfo(x) <- value

## S4 replacement method for signature 'Analysis'
sinfo(x, type = c("pre-treated", "raw")) <- value

raw(x)

## S4 method for signature 'Analysis'
raw(x)

raw(x) <- value

## S4 replacement method for signature 'Analysis'
raw(x) <- value

preTreated(x)

## S4 method for signature 'Analysis'
preTreated(x)

preTreated(x) <- value

## S4 replacement method for signature 'Analysis'
preTreated(x) <- value

features(x, ...)

## S4 method for signature 'AnalysisData'
features(x)

## S4 method for signature 'Analysis'
features(x, type = c("pre-treated", "raw"))

nSamples(x, ...)
```

```
## S4 method for signature 'AnalysisData'
nSamples(x)

## S4 method for signature 'Analysis'
nSamples(x, type = c("pre-treated", "raw"))

nFeatures(x, ...)

## S4 method for signature 'AnalysisData'
nFeatures(x)

## S4 method for signature 'Analysis'
nFeatures(x, type = c("pre-treated", "raw"))

analysisResults(x, element)

## S4 method for signature 'Analysis'
analysisResults(x, element)
```

Arguments

x	S4 object of class AnalysisData or Analysis
...	arguments to pass to the appropriate method
type	get or set raw or pre-treated data
value	value to set
element	analysis element results to return

Methods

- dat: Return a metabolomic data table.
- dat<-: Set a metabolomic data table.
- sinfo: Return a sample information data table.
- sinfo<-: Set a sample information data table.
- raw: Return the AnalysisData object containing unprocessed metabolomic data from an Analysis object.
- raw<-: Set an AnalysisData object to the raw slot of an Analysis class object.
- preTreated: Return the AnalysisData object containing pre-treated metabolomic data from an Analysis object.
- preTreated<-: Set an AnalysisData object to the pre-treated slot of an Analysis class object.
- features: Return the features names.
- nSamples: Return the number of samples.
- nFeatures: Return the number of features.
- analysisResults: Return results from an Analysis object of an analysis element.

Examples

```
library(metaboData)

d <- analysisData(abr1$neg[,200:300],abr1$fact)

## Return the metabolomic data
dat(d)

## Set the metabolomic data
dat(d) <- abr1$neg[,300:400]

## Return the sample information
sinfo(d)

## Set the sample information
sinfo(d) <- abr1$fact

## Return the feature names
features(d)

## Return the number of samples
nSamples(d)

## Return the number of features
nFeatures(d)
```

`imputeAll` *Missing data imputation*

Description

Impute missing values using random forest imputation.

Usage

```
imputeAll(d, occupancy = 2/3, parallel = "variables", seed = 1234)

## S4 method for signature 'AnalysisData'
imputeAll(d, occupancy = 2/3, parallel = "variables", seed = 1234)

imputeClass(d, cls = "class", occupancy = 2/3, seed = 1234)

## S4 method for signature 'AnalysisData'
imputeClass(d, cls = "class", occupancy = 2/3, seed = 1234)
```

Arguments

`d`	S4 object of class `AnalysisData`

occupancy	occupancy threshold above which missing values of a feature will be imputed
parallel	parallel type to use. See ?missForest for details
seed	random number seed
cls	info column to use for class labels

Details

Missing values can have an important influence on downstream analyses with zero values heavily influencing the outcomes of parametric tests. Where and how they are imputed are important considerations and is highly related to variable occupancy. The methods provided here allow both these aspects to be taken into account and utilise random forest imputation using the missForest package.

Value

An S4 object of class AnalysisData containing the data after imputation.

Methods

- imputeAll: Impute missing values across all sample features.
- imputeClass: Impute missing values class-wise.

Examples

```
## Each of the following examples shows the application of each imputation method and then
## a Linear Discriminant Analysis is plotted to show it's effect on the data structure.

## Initial example data preparation
library(metaboData)

d <- analysisData(abr1$neg[,200:250],abr1$fact) %>%
 occupancyMaximum(occupancy = 2/3)

d %>%
 plotLDA(cls = 'day')

## Missing value imputation across all samples
d %>%
 imputeAll(parallel = 'no') %>%
 plotLDA(cls = 'day')

## Missing value imputation class-wise
d %>%
 imputeClass(cls = 'day') %>%
 plotLDA(cls = 'day')
```

keepClasses *Keep samples, classes or features*

Description

Retain samples, classes or features in an AnalysisData object.

Usage

```
keepClasses(d, cls = "class", classes = c())

## S4 method for signature 'AnalysisData'
keepClasses(d, cls = "class", classes = c())

keepFeatures(d, features = character())

## S4 method for signature 'AnalysisData'
keepFeatures(d, features = character())

keepSamples(d, idx = "fileOrder", samples = c())

## S4 method for signature 'AnalysisData'
keepSamples(d, idx = "fileOrder", samples = c())
```

Arguments

d	S4 object of class AnalysisData
cls	info column to use for class information
classes	classes to keep
features	features to remove
idx	info column containing sample indexes
samples	sample indexes to keep

Value

An S4 object of class AnalysisData with specified samples, classes or features retained.

Methods

- keepClasses: Keep classes.
- keepFeatures: Keep features.
- keepSamples: Keep samples.

Examples

```
library(metaboData)
 d <- analysisData(abr1$neg[,200:300],abr1$fact)

 ## Keep classes
 d %>%
  keepClasses(cls = 'day',classes = 'H')

 ## Keep features
 d %>%
  keepFeatures(features = c('N200','N201'))

 ## Keep samples
 d %>%
  keepSamples(idx = 'injorder',samples = c(1,10))
```

`linearRegression` *Linear regression*

Description

Linear regression

Usage

```
linearRegression(
  x,
  cls = "class",
  pAdjust = "bonferroni",
  returnModels = FALSE
)

## S4 method for signature 'AnalysisData'
linearRegression(
  x,
  cls = "class",
  pAdjust = "bonferroni",
  returnModels = FALSE
)
```

Arguments

`x`	S4 object of class `AnalysisData`
`cls`	vector of sample information column names to regress
`pAdjust`	p value adjustment method
`returnModels`	should models be returned

Value

An S4 object of class `Univariate`.

Examples

```
library(metaboData)

d <- analysisData(abr1$neg[,200:300],abr1$fact)

## Perform linear regression
lr_analysis <- linearRegression(d,cls = 'injorder')

## Extract significant features
explanatoryFeatures(lr_analysis)
```

`mds` *Multidimensional scaling (MDS)*

Description

Multidimensional scaling of random forest proximities.

Usage

```
mds(x, dimensions = 2, idx = NULL)

## S4 method for signature 'RandomForest'
mds(x, dimensions = 2, idx = NULL)

## S4 method for signature 'list'
mds(x, dimensions = 2, idx = NULL)

## S4 method for signature 'Analysis'
mds(x, dimensions = 2, idx = NULL)
```

Arguments

`x`	S4 object of class `RandomForest`, `Analysis` or a list
`dimensions`	The number of dimensions by which the data are to be represented.
`idx`	sample information column to use for sample names. If `NULL`, the sample row number will be used. Sample names should be unique for each row of data.

Value

A tibble containing the scaled dimensions.

Examples

```
library(metaboData)

x <- analysisData(abr1$neg[,200:300],abr1$fact) %>%
       occupancyMaximum(cls = 'day') %>%
       transformTICnorm()

rf <- randomForest(x,cls = 'day')

mds(rf)
```

`metabolyse` *Perform an analysis*

Description

Perform analyses containing multiple analysis element steps.

Usage

```
metabolyse(data, info, parameters = analysisParameters(), verbose = TRUE)

reAnalyse(analysis, parameters = analysisParameters(), verbose = TRUE)

## S4 method for signature 'Analysis'
reAnalyse(analysis, parameters = analysisParameters(), verbose = TRUE)
```

Arguments

`data`	tibble or data.frame containing data to analyse
`info`	tibble or data.frame containing data info or meta data
`parameters`	an object of AnalysisParameters class containing parameters for analysis. Default calls `analysisParameters()`
`verbose`	should output be printed to the console
`analysis`	an object of class Analysis containing previous analysis results

Details

Routine analyses are those that are often made up of numerous steps where parameters have likely already been previously established. The emphasis here is on convenience with as little code as possible required. In these analyses, the necessary analysis elements, order and parameters are first prepared and then the analysis routine subsequently performed in a single step. The `metabolyse` function provides this utility, where the metabolome data, sample meta information and analysis parameters are provided. The `reAnalyse` method can be used to perform further analyses on the results.

Value

An S4 object of class `Analysis`.

Examples

```
library(metaboData)

## Generate analysis parameters
p <- analysisParameters(c('pre-treatment','modelling'))

## Alter pre-treatment and modelling parameters to use different methods
parameters(p,'pre-treatment') <- preTreatmentParameters(
  list(occupancyFilter = 'maximum',
       transform = 'TICnorm')
)
parameters(p,'modelling') <- modellingParameters('anova')

## Change "cls" parameters
changeParameter(p,'cls') <- 'day'

## Run analysis using a subset of the abr1 negative mode data set
analysis <- metabolyse(abr1$neg[,1:200],
                       abr1$fact,
                       p)

## Re-analyse to include correlation analysis
analysis <- reAnalyse(analysis,
                      parameters = analysisParameters('correlations'))

print(analysis)
```

---

modellingMethods *Modelling parameters*

Description

Retrieve the available modelling methods and parameters.

Usage

```
modellingMethods()

modellingParameters(methods)
```

Arguments

`methods`	character vector of available modelling methods

Examples

```
## Retrieve the available modelling methods
modellingMethods()

## Retrieve the modelling parameters for the anova method
p <- modellingParameters('anova')

## Assign the modelling parameters to analysis parameters
mp <- analysisParameters('modelling')

parameters(mp,'modelling') <- p

print(mp)
```

---

occupancy *Calculate feature class occupancies*

Description

Calculate the class occupancies of all features in an `AnalysisData` object.

Usage

```
occupancy(d, cls = "class")

## S4 method for signature 'AnalysisData'
occupancy(d, cls = "class")
```

Arguments

`d`	S4 object of class `AnalysisData`
`cls`	sample information column to use for which to compute class occupancies

Value

A tibble containing feature class proportional occupancies.

Examples

```
library(metaboData)

d <- analysisData(abr1$neg[,200:300],abr1$fact)

occupancy(d,cls = 'day')
```

occupancyMaximum *Feature occupancy filtering*

Description

Feature filtering based on class occupancy.

Usage

```
occupancyMaximum(d, cls = "class", occupancy = 2/3)

## S4 method for signature 'AnalysisData'
occupancyMaximum(d, cls = "class", occupancy = 2/3)

occupancyMinimum(d, cls = "class", occupancy = 2/3)

## S4 method for signature 'AnalysisData'
occupancyMinimum(d, cls = "class", occupancy = 2/3)
```

Arguments

d	S4 object of class AnalysisData
cls	sample information column name to use for class data
occupancy	feature occupancy filtering threshold, below which features will be removed

Details

Occupancy provides a useful metric by which to filter poorly represented features (features containing a majority zero or missing values). An occupancy threshold provides a means of specifying this majority with variables below the threshold excluded from further analyses. However, this can be complicated by an underlying class structure present within the data where a variable may be well represented within one class but not in another.

Value

An S4 object of class AnalysisData containing the class occupancy filtered data.

Methods

- occupancyMaximium: Maximum occupancy threshold feature filtering. Where the maximum occupancy across all classes is above the threshold. Therefore, for a feature to be retained, only a single class needs to have an occupancy above the threshold.
- occupancyMinimum: Minimum occupancy threshold feature filtering. Where the minimum occupancy across all classes is required to be above the threshold. Therefore, for a feature to be retained, all classes would need to have an occupancy above the threshold.

Examples

```
## Each of the following examples shows the application
## of the feature occupancy filtering method method and
## then a Principle Component Analysis is plotted to show
## its effect on the data structure.

## Initial example data preparation
library(metaboData)

d <- analysisData(abr1$neg[,200:300],abr1$fact)

## Maximum occupancy threshold feature filtering
d %>%
 occupancyMaximum(cls = 'day') %>%
 plotPCA(cls = 'day')

## Minimum occupancy threshold feature filtering
d %>%
 occupancyMinimum(cls = 'day') %>%
 plotPCA(cls = 'day')
```

`parameters` *Get or set analysis parameters*

Description

Get or set parameters for `AnalysisParameters` or `Analysis` class objects.

Usage

```
parameters(d, ...)

## S4 method for signature 'AnalysisParameters'
parameters(d, element)

## S4 method for signature 'Analysis'
parameters(d)

parameters(d, element) <- value

## S4 replacement method for signature 'AnalysisParameters'
parameters(d, element) <- value

## S4 replacement method for signature 'Analysis'
parameters(d) <- value
```

Arguments

d	S4 object of class AnalysisParameters or Analysis
...	arguments to pass to the appropriate method
element	analysis element for parameters to extract or assign. Should be one of those returned by analysisElements()
value	list containing parameter values

Examples

```
p <- analysisParameters('pre-treatment')

## extract pre-treatment parameters
parameters(p,'pre-treatment')

## set pre-treatment parameters
parameters(p,'pre-treatment') <- preTreatmentParameters(
  list(
    remove = 'classes',
    QC = c('RSDfilter','removeQC'),
    transform = 'TICnorm'
  )
)

print(p)
```

---

parseParameters *Parse/export analysis parameters*

Description

Import analysis parameters from a .yaml format file or export an AnalysisParameters object to .yaml format.

Usage

```
parseParameters(path)

exportParameters(d, file = "analysis_parameters.yaml")

## S4 method for signature 'AnalysisParameters'
exportParameters(d, file = "analysis_parameters.yaml")

## S4 method for signature 'Analysis'
exportParameters(d, file = "analysis_parameters.yaml")
```

Arguments

path	file path of .yaml file to parse
d	S4 object of class AnalysisParameters or Analysis
file	File name and path to export to

Examples

```
## Import analysis parameters
paramFile <- system.file('defaultParameters.yaml',package = 'metabolyseR')
p <- parseParameters(paramFile)
p

## Not run:
## Export analysis parameters
exportParameters(p,file = 'analysis_parameters.yaml')

## End(Not run)
```

---

plotExplanatoryHeatmap

Heatmap plot of explantory features

Description

Plot a heatmap of explanatory features.

Usage

```
plotExplanatoryHeatmap(x, ...)

## S4 method for signature 'Univariate'
plotExplanatoryHeatmap(
  x,
  threshold = 0.05,
  title = "",
  distanceMeasure = "euclidean",
  clusterMethod = "ward.D2",
  featureNames = TRUE,
  dendrogram = TRUE,
  featureLimit = Inf,
  ...
)

## S4 method for signature 'RandomForest'
plotExplanatoryHeatmap(
  x,
  metric = "false_positive_rate",
```

```
  threshold = 0.05,
  title = "",
  distanceMeasure = "euclidean",
  clusterMethod = "ward.D2",
  featureNames = TRUE,
  dendrogram = TRUE,
  featureLimit = Inf,
  ...
)

## S4 method for signature 'list'
plotExplanatoryHeatmap(
  x,
  threshold = 0.05,
  distanceMeasure = "euclidean",
  clusterMethod = "ward.D2",
  featureNames = TRUE,
  featureLimit = Inf
)

## S4 method for signature 'Analysis'
plotExplanatoryHeatmap(
  x,
  threshold = 0.05,
  distanceMeasure = "euclidean",
  clusterMethod = "ward.D2",
  featureNames = TRUE,
  featureLimit = Inf
)
```

Arguments

x	object of class Univariate, RandomForest or Analysis
...	arguments to pass to method explanatoryFeatures()
threshold	score threshold to use for specifying explanatory features
title	plot title
distanceMeasure	distance measure to use for clustering. See details.
clusterMethod	clustering method to use. See details
featureNames	should feature names be plotted?
dendrogram	TRUE/FALSE. Should the dendrogram be plotted?
featureLimit	The maximum number of features to plot
metric	importance metric on which to retrieve explanatory features

Details

Distance measures can be one of any that can be used for the method argument of dist().

Cluster methods can be one of any that can be used for the method argument of hclust().

Examples

```
library(metaboData)
x <- analysisData(data = abr1$neg[,200:300],info = abr1$fact)

## random forest classification example
random_forest <- randomForest(x,cls = 'day')

plotExplanatoryHeatmap(random_forest)

## random forest regression example
random_forest <- randomForest(x,cls = 'injorder')

plotExplanatoryHeatmap(random_forest,metric = '%IncMSE',threshold = 2)
```

---

plotFeature *Plot a feature*

Description

Plot the trend of a feature.

Usage

```
plotFeature(analysis, feature, cls = "class", label = NULL, labelSize = 2, ...)

## S4 method for signature 'AnalysisData'
plotFeature(analysis, feature, cls = "class", label = NULL, labelSize = 2)

## S4 method for signature 'Analysis'
plotFeature(
  analysis,
  feature,
  cls = "class",
  label = NULL,
  labelSize = 2,
  type = c("pre-treated", "raw")
)
```

Arguments

analysis	an object of class AnalysisData or“ Analysis‘
feature	feature name to plot
cls	information column to use for class labels
label	information column to use for sample labels

labelSize	sample label size
...	arguments to pass to the appropriate method
type	raw or pre-treated data to plot

Examples

```
d <- analysisData(metaboData::abr1$neg,
                  metaboData::abr1$fact)

## Plot a categorical response variable
plotFeature(d,'N133',cls = 'day')

## Plot a continuous response variable
plotFeature(d,'N133',cls = 'injorder')
```

---

plotImportance *Plot feature importance*

Description

Plot Univariate or random forest feature importance.

Usage

```
plotImportance(x, ...)

## S4 method for signature 'Univariate'
plotImportance(x, response = "class", rank = TRUE, threshold = 0.05)

## S4 method for signature 'RandomForest'
plotImportance(x, metric = "false_positive_rate", rank = TRUE)

## S4 method for signature 'list'
plotImportance(x, metric = "false_positive_rate")
```

Arguments

x	S4 object of class Univariate or RandomForest
...	arguments to pass to specific method
response	response results to plot
rank	rank feature order for plotting
threshold	explanatory threshold line for the output plot
metric	importance metric to plot

Examples

```
library(metaboData)

x <- analysisData(abr1$neg[,200:300],abr1$fact) %>%
       keepClasses(cls = 'day',classes = c('H','1','5')) %>%
       occupancyMaximum(cls = 'day') %>%
       transformTICnorm()

rf <- randomForest(x,cls = 'day')

plotImportance(rf,rank = FALSE)
```

---

plotLDA *Principle Component - Linear Discriminant Analysis plot*

Description

Plot linear discriminant analysis results of pre-treated data

Usage

```
plotLDA(
  analysis,
  cls = "class",
  label = NULL,
  scale = TRUE,
  center = TRUE,
  xAxis = "DF1",
  yAxis = "DF2",
  shape = FALSE,
  ellipses = TRUE,
  title = "PC-LDA",
  legendPosition = "bottom",
  labelSize = 2,
  ...
)

## S4 method for signature 'AnalysisData'
plotLDA(
  analysis,
  cls = "class",
  label = NULL,
  scale = TRUE,
  center = TRUE,
  xAxis = "DF1",
  yAxis = "DF2",
  shape = FALSE,
```

```
  ellipses = TRUE,
  title = "PC-LDA",
  legendPosition = "bottom",
  labelSize = 2
)

## S4 method for signature 'Analysis'
plotLDA(
  analysis,
  cls = "class",
  label = NULL,
  scale = TRUE,
  center = TRUE,
  xAxis = "DF1",
  yAxis = "DF2",
  shape = FALSE,
  ellipses = TRUE,
  title = "PC-LDA",
  legendPosition = "bottom",
  labelSize = 2,
  type = c("pre-treated", "raw")
)
```

Arguments

analysis	S4 object of class AnalysisData or Analysis
cls	name of sample information column to use for class labels
label	name of sample information column to use for sample labels. Set to NULL for no labels.
scale	scale the data
center	center the data
xAxis	principle component to plot on the x-axis
yAxis	principle component to plot on the y-axis
shape	TRUE/FALSE use shape aesthetic for plot points. Defaults to TRUE when the number of classes is greater than 12
ellipses	TRUE/FALSE, plot multivariate normal distribution 95\ confidence ellipses for each class
title	plot title
legendPosition	legend position to pass to legend.position argument of ggplot2::theme. Set to "none" to remove legend.
labelSize	label size. Ignored if label is NULL
...	arguments to pass to the appropriate method
type	raw or pre-treated data to plot

Examples

```
library(metaboData)

d <- analysisData(abr1$neg,abr1$fact) %>%
 occupancyMaximum(cls = 'day')

## LDA plot
plotLDA(d,cls = 'day')
```

---

plotMDS *Multidimensional scaling (MDS) plot*

Description

Plot multidimensional scaling plot for a RandomForest class object.

Usage

```
plotMDS(
  x,
  cls = "class",
  label = NULL,
  shape = FALSE,
  ellipses = TRUE,
  title = "",
  legendPosition = "bottom",
  labelSize = 2
)

## S4 method for signature 'RandomForest'
plotMDS(
  x,
  cls = "class",
  label = NULL,
  shape = FALSE,
  ellipses = TRUE,
  title = "",
  legendPosition = "bottom",
  labelSize = 2
)

## S4 method for signature 'list'
plotMDS(
  x,
  label = NULL,
  shape = FALSE,
  ellipses = TRUE,
```

```
  title = "",
  legendPosition = "bottom",
  labelSize = 2
)
```

Arguments

x	S4 object of class RandomForest
cls	sample information column to use for sample labelling, Set to NULL for no labelling.
label	sample information column to use for sample labels. Set to NULL for no labels.
shape	TRUE/FALSE use shape aesthetic for plot points. Defaults to TRUE when the number of classes is greater than 12
ellipses	TRUE/FALSE, plot multivariate normal distribution 95% confidence ellipses for each class
title	plot title
legendPosition	legend position to pass to legend.position argument of ggplot2::theme. Set to "none" to remove legend.
labelSize	label size. Ignored if label is NULL

Examples

```
library(metaboData)

x <- analysisData(abr1$neg[,200:300],abr1$fact) %>%
       occupancyMaximum(cls = 'day') %>%
       transformTICnorm()

rf <- randomForest(x,cls = 'day')

plotMDS(rf,cls = 'day')
```

---

plotMetrics *Plot model performance metrics*

Description

Plot random forest model performance metrics

Usage

```
plotMetrics(x, response = "class")

## S4 method for signature 'RandomForest'
plotMetrics(x)

## S4 method for signature 'list'
plotMetrics(x)
```

Arguments

x	S4 object of class RandomForest
response	response results to plot

Examples

```
library(metaboData)

x <- analysisData(abr1$neg[,200:300],abr1$fact) %>%
       keepClasses(cls = 'day',classes = c('H','1','5')) %>%
       occupancyMaximum(cls = 'day') %>%
       transformTICnorm()

rf <- randomForest(x,cls = 'day',binary = TRUE)

plotMetrics(rf,response = 'day')
```

---

plotOccupancy *Plot class occupancy distributions*

Description

Plot class occupancy distributions.

Usage

```
plotOccupancy(x, cls = "class", ...)

## S4 method for signature 'AnalysisData'
plotOccupancy(x, cls = "class")

## S4 method for signature 'Analysis'
plotOccupancy(x, cls = "class", type = "raw")
```

Arguments

x	S4 object of class AnalysisData or Analysis
cls	sample information column to use for class labels
...	arguments to pass to the appropriate method
type	raw or preTreated data to plot

Examples

```
library(metaboData)

d <- analysisData(abr1$neg,abr1$fact)

## Plot class occupancy distributions
plotOccupancy(d,cls = 'day')
```

plotPCA *Principle Component Analysis plot*

Description

Plot Principle Component Analysis results.

Usage

```
plotPCA(
  analysis,
  cls = "class",
  label = NULL,
  scale = TRUE,
  center = TRUE,
  xAxis = "PC1",
  yAxis = "PC2",
  shape = FALSE,
  ellipses = TRUE,
  title = "PCA",
  legendPosition = "bottom",
  labelSize = 2,
  ...
)

## S4 method for signature 'AnalysisData'
plotPCA(
  analysis,
  cls = "class",
  label = NULL,
  scale = TRUE,
  center = TRUE,
  xAxis = "PC1",
  yAxis = "PC2",
  shape = FALSE,
  ellipses = TRUE,
  title = "Principle Component Analysis (PCA)",
  legendPosition = "bottom",
  labelSize = 2
)

## S4 method for signature 'Analysis'
plotPCA(
  analysis,
  cls = "class",
  label = NULL,
  scale = TRUE,
```

```
  center = TRUE,
  xAxis = "PC1",
  yAxis = "PC2",
  shape = FALSE,
  ellipses = TRUE,
  title = "PCA",
  legendPosition = "bottom",
  labelSize = 2,
  type = c("pre-treated", "raw")
)
```

Arguments

analysis	object of class AnalysisData or Analysis
cls	name of class information column to use for sample labelling
label	name of class information column to use for sample labels. Set to NULL for no labels.
scale	scale the data
center	center the data
xAxis	principle component to plot on the x-axis
yAxis	principle component to plot on the y-axis
shape	TRUE/FALSE use shape aesthetic for plot points. Defaults to TRUE when the number of classes is greater than 12
ellipses	TRUE/FALSE, plot multivariate normal distribution 95\ confidence ellipses for each class
title	plot title
legendPosition	legend position to pass to legend.position argument of ggplot2::theme. Set to "none" to remove legend.
labelSize	label size. Ignored if label is NULL
...	arguments to pass to the appropriate method
type	raw or pre-treated data to plot

Examples

```
library(metaboData)

d <- analysisData(abr1$neg,abr1$fact) %>%
 occupancyMaximum(cls = 'day')

## PCA plot
plotPCA(d,cls = 'day')
```

plotROC *Plot receiver operator characteristic (ROC) curves*

Description

Plot receiver operator characteristic curves for a RandomForest class object.

Usage

```
plotROC(x, title = "", legendPosition = "bottom")

## S4 method for signature 'RandomForest'
plotROC(x, title = "", legendPosition = "bottom")

## S4 method for signature 'list'
plotROC(x, title = "", legendPosition = "bottom")
```

Arguments

x	S4 object of class RandomForest
title	plot title
legendPosition	legend position to pass to legend.position argument of ggplot2::theme. Set to "none" to remove legend.

Examples

```
library(metaboData)

x <- analysisData(abr1$neg[,200:300],abr1$fact) %>%
       occupancyMaximum(cls = 'day') %>%
       transformTICnorm()

rf <- randomForest(x,cls = 'day')

plotROC(rf)
```

plotRSD *Plot RSD distributions*

Description

Plot RSD distributions of raw data in quality control samples.

Usage

```
plotRSD(analysis, cls = "class", ...)

## S4 method for signature 'AnalysisData'
plotRSD(analysis, cls = "class")

## S4 method for signature 'Analysis'
plotRSD(analysis, cls = "class", type = "raw")
```

Arguments

analysis	object of class AnalysisData or Analysis
cls	information column to use for class labels
...	arguments to pass to the appropriate method
type	raw or pre-treated data to plot

Examples

```
library(metaboData)

d <- analysisData(abr1$neg,abr1$fact)

## Plot class RSD distributions
plotRSD(d,cls = 'day')
```

---

plotSupervisedRF *Supervised random forest MDS plot*

Description

A multidimensional scaling (MDS) plot of supervised random forest analysis

Usage

```
plotSupervisedRF(
  x,
  cls = "class",
  rf = list(),
  label = NULL,
  shape = FALSE,
  ellipses = TRUE,
  ROC = TRUE,
  seed = 1234,
  title = "",
  legendPosition = "bottom",
  labelSize = 2,
```

```
  ...
)

## S4 method for signature 'AnalysisData'
plotSupervisedRF(
  x,
  cls = "class",
  rf = list(),
  label = NULL,
  shape = FALSE,
  ellipses = TRUE,
  ROC = TRUE,
  seed = 1234,
  title = "",
  legendPosition = "bottom",
  labelSize = 2
)

## S4 method for signature 'Analysis'
plotSupervisedRF(
  x,
  cls = "class",
  rf = list(),
  label = NULL,
  shape = FALSE,
  ellipses = TRUE,
  ROC = TRUE,
  seed = 1234,
  title = "",
  legendPosition = "bottom",
  labelSize = 2,
  type = c("pre-treated", "raw")
)
```

Arguments

x	object of class AnalysisData or Analysis containing analysis results
cls	information column to use for sample classes
rf	list of additional parameters to pass to randomForest
label	information column to use for sample labels. Set to NULL for no labels.
shape	TRUE/FALSE use shape aesthetic for plot points. Defaults to TRUE when the number of classes is greater than 12
ellipses	TRUE/FALSE, plot multivariate normal distribution 95% confidence ellipses for each class
ROC	should receiver-operator characteristics be plotted?
seed	random number seed

title	plot title
legendPosition	legend position to pass to legend.position argument of ggplot2::theme. Set to "none" to remove legend.
labelSize	label size. Ignored if label is NULL
...	arguments to pass to the appropriate method
type	raw or pre-treated data to plot

Examples

```
library(metaboData)

d <- analysisData(abr1$neg[,200:300],abr1$fact)

## Supervised random forest MDS plot
plotSupervisedRF(d,cls = 'day')
```

---

plotTIC *Plot sample total ion counts*

Description

Plot total ion counts of sample data.

Usage

```
plotTIC(analysis, by = "injOrder", colour = "block", ...)

## S4 method for signature 'AnalysisData'
plotTIC(analysis, by = "injOrder", colour = "block")

## S4 method for signature 'Analysis'
plotTIC(
  analysis,
  by = "injOrder",
  colour = "block",
  type = c("pre-treated", "raw")
)
```

Arguments

analysis	S4 object of class AnalysisData or Analysis
by	information column to plot against
colour	information column to provide colour labels
...	arguments to pass to the appropriate method
type	raw or pre-treated sample data

Examples

```
library(metaboData)

d <- analysisData(abr1$neg,abr1$fact)

## Plot sample TIVs
plotTIC(d,by = 'injorder',colour = 'day')

plotTIC(d,by = 'day',colour = 'day')
```

---

plotUnsupervisedRF *Unsupervised random forest MDS plot*

Description

A multidimensional scaling (MDS) plot of unsupervised random forest analysis

Usage

```
plotUnsupervisedRF(
  x,
  cls = "class",
  rf = list(),
  label = NULL,
  shape = FALSE,
  ellipses = TRUE,
  seed = 1234,
  title = "",
  legendPosition = "bottom",
  labelSize = 2,
  ...
)

## S4 method for signature 'AnalysisData'
plotUnsupervisedRF(
  x,
  cls = "class",
  rf = list(),
  label = NULL,
  shape = FALSE,
  ellipses = TRUE,
  seed = 1234,
  title = "",
  legendPosition = "bottom",
  labelSize = 2
)
```

```
## S4 method for signature 'Analysis'
plotUnsupervisedRF(
  x,
  cls = "class",
  rf = list(),
  label = NULL,
  shape = FALSE,
  ellipses = TRUE,
  seed = 1234,
  title = "",
  legendPosition = "bottom",
  labelSize = 2,
  type = c("pre-treated", "raw")
)
```

Arguments

x	object of class AnalysisData or Analysis
cls	sample information column to use for sample labelling
rf	list of additional parameters to pass to randomForest
label	info column to use for sample labels. Set to NULL for no labels.
shape	TRUE/FALSE use shape aesthetic for plot points. Defaults to TRUE when the number of classes is greater than 12
ellipses	TRUE/FALSE, plot multivariate normal distribution 95% confidence ellipses for each class
seed	random number seed
title	plot title
legendPosition	legend position to pass to legend.position argument of ggplot2::theme. Set to "none" to remove legend.
labelSize	label size. Ignored if label is NULL
...	arguments to pass to the appropriate method
type	raw or pre-treated data to plot

Examples

```
library(metaboData)

d <- analysisData(abr1$neg[,200:300],abr1$fact)

## Unsupervised random forest MDS plot
plotUnsupervisedRF(d,cls = 'day')
```

predict *Predict random forest model responses*

Description

Predict values of random forest model response variables from new data.

Usage

```
predict(
  model,
  new_data,
  idx = NULL,
  type = c("response", "prob", "votes"),
  ...
)

## S4 method for signature 'RandomForest,AnalysisData'
predict(
  model,
  new_data,
  idx = NULL,
  type = c("response", "prob", "votes"),
  ...
)
```

Arguments

model	S4 object of class RandomForest
new_data	S4 object of class AnalysisData
idx	sample information column to use for sample names. If NULL, the sample row number will be used. Sample names should be unique for each row of data.
type	one of response, prob, or votes to indicate the type of prediction to make
...	arguments to pass to randomForest::predict.randomForest()

Details

The features contained within new_data should match those of the features used to train model. The features() method can be used to check this. The argument returnModels = TRUE should also be used when training the RandomForest-class object used for argument model.

Examples

```
library(metaboData)

## Prepare some data
```

```
x <- analysisData(abr1$neg[,200:300],abr1$fact) %>%
  occupancyMaximum(cls = 'day') %>%
  transformTICnorm()

## Extract data from which to train a random forest model
training_data <- x %>%
  keepClasses(cls = 'day',
              classes = c('H','1'))

## Extract data for which response values will be predicted
test_data <- x %>%
  keepClasses(cls = 'day',
              classes = c('2','3'))

rf <- randomForest(training_data,
                   cls = 'day',
                   returnModels = TRUE)

predict(rf,
        test_data)
```

`preTreatmentElements` *Pre-treatment parameters*

Description

Return pre-treatment elements, methods and parameters.

Usage

```
preTreatmentElements()

preTreatmentMethods(element)

preTreatmentParameters(methods)
```

Arguments

`element`	pre-treatment element name
`methods`	a named list of element methods

Examples

```
## Return the availalble pre-treatment elements
preTreatmentElements()

## Return the available pre-treatment methods for the remove element
preTreatmentMethods('remove')
```

```
## Define some default pre-treatment parameters
p <- preTreatmentParameters(
  list(
    remove = 'classes',
    QC = c('RSDfilter','removeQC'),
    transform = 'TICnorm'
  )
)

## Assign the pre-treatment parameters to analysis parameters
ap <- analysisParameters('pre-treatment')
parameters(ap,'pre-treatment') <- p

print(ap)
```

QCimpute *Quality control (QC) sample treatments*

Description

Quality control (QC) sample pre-treatment methods.

Usage

```
QCimpute(
  d,
  cls = "class",
  QCidx = "QC",
  occupancy = 2/3,
  parallel = "variables",
  seed = 1234
)

## S4 method for signature 'AnalysisData'
QCimpute(
  d,
  cls = "class",
  QCidx = "QC",
  occupancy = 2/3,
  parallel = "variables",
  seed = 1234
)

QCoccupancy(d, cls = "class", QCidx = "QC", occupancy = 2/3)

## S4 method for signature 'AnalysisData'
QCoccupancy(d, cls = "class", QCidx = "QC", occupancy = 2/3)
```

```
QCremove(d, cls = "class", QCidx = "QC")

## S4 method for signature 'AnalysisData'
QCremove(d, cls = "class", QCidx = "QC")

QCrsdFilter(d, cls = "class", QCidx = "QC", RSDthresh = 50)

## S4 method for signature 'AnalysisData'
QCrsdFilter(d, cls = "class", QCidx = "QC", RSDthresh = 50)
```

Arguments

d	S4 object of class AnalysisData
cls	info column to use for class labels
QCidx	QC sample label
occupancy	occupancy threshold for filtering
parallel	parallel type to use. See ?missForest for details
seed	random number seed
RSDthresh	RSD (%) threshold for filtering

Details

A QC sample is an average pooled sample, equally representative in composition of all the samples present within an experimental set. Within an analytical run, the QC sample is analysed at equal intervals throughout the run. If there is class structure within the run, this should be randomised within a block fashion so that the classes are equally represented in each block throughout the run. A QC sample can then be injected and analysed between these randomised blocks. This provides a set of technical injections that allows the variability in instrument performance over the run to be accounted for and the robustness of the acquired variables to be assessed.

The technical reproducibility of an acquired variable can be assessed using it's relative standard deviation (RSD) within the QC samples. The variable RSDs can then be filtered below a threshold value to remove metabolome features that are poorly reproducible across the analytical runs. This variable filtering strategy has an advantage over that of occupancy alone as it is not dependent on underlying class structure. Therefore, the variables and variable numbers will not alter if a new class structure is imposed upon the data.

Value

An S4 object of class `AnalysisData` containing QC treated data.

Methods

- `QCimpute`: Missing value imputation of QC samples.
- `QCoccupancy`: Feature maximum occupancy filtering based on QC samples.
- `QCremove`: Remove QC samples.
- `QCrsdFilter`: Feature filtering based RSD of QC sample features.

Examples

```
## Initial example data preparation
library(metaboData)
d <- analysisData(abr1$neg[,1:1000],abr1$fact)

## Plot the feature RSD distributions of the H class only
d %>%
 keepClasses(cls = 'day',classes = 'H') %>%
 plotRSD(cls = 'day')

## Apply QC feature occupancy filtering and QC feature RSD filtering
QC_treated <- d %>%
 QCoccupancy(cls = 'day',QCidx = 'H',occupancy = 2/3) %>%
 QCrsdFilter(cls = 'day',QCidx = 'H',RSDthresh = 50)

print(QC_treated)

## Plot the feature RSD distributions of the H class after QC treatments
QC_treated %>%
 keepClasses(cls = 'day',classes = 'H') %>%
 plotRSD(cls = 'day')
```

`randomForest` *Random forest*

Description

Perform random forest on an `AnalysisData` object

Usage

```
randomForest(
  x,
  cls = "class",
  rf = list(),
  reps = 1,
  binary = FALSE,
  comparisons = list(),
  perm = 0,
  returnModels = FALSE,
  seed = 1234
)

## S4 method for signature 'AnalysisData'
randomForest(
  x,
  cls = "class",
  rf = list(),
```

```
  reps = 1,
  binary = FALSE,
  comparisons = list(),
  perm = 0,
  returnModels = FALSE,
  seed = 1234
)
```

Arguments

x	S4 object of class AnalysisData
cls	vector of sample information columns to use for response variable information. Set to NULL for unsupervised.
rf	named list of arguments to pass to randomForest::randomForest
reps	number of repetitions to perform
binary	TRUE/FALSE should binary comparisons be performed. Ignored for unsupervised and regression. Ignored if comparisons specified.
comparisons	list of comparisons to perform. Ignored for unsupervised and regression. See details.
perm	number of permutations to perform. Ignored for unsupervised.
returnModels	TRUE/FALSE should model objects be returned.
seed	random number seed

Details

Specified class comparisons should be given as a list named according to cls. Comparisons should be given as class names separated by '~' (eg. '1~2~H').

Value

An S4 object of class RandomForest.

Examples

```
library(metaboData)

x <- analysisData(abr1$neg[,200:300],abr1$fact) %>%
       occupancyMaximum(cls = 'day') %>%
       transformTICnorm()

rf <- randomForest(x,cls = 'day')

plotMDS(rf,cls = 'day')
```

RandomForest-class *RandomForest S4 class*

Description

An S4 class for random forest results and models.

Slots

`type` random forest type

`response` response variable name

`metrics` tibble of model performance metrics

`predictions` tibble of model observation predictions

`permutations` list of permutations measure and importance results tables

`importances` tibble of model feature importances

`proximities` tibble of model observation proximities

`models` list of random forest models

removeClasses *Remove samples, classes or features*

Description

Exclusion of samples, classes or features from an `AnalysisData` object.

Usage

```
removeClasses(d, cls = "class", classes = c())

## S4 method for signature 'AnalysisData'
removeClasses(d, cls = "class", classes = c())

removeFeatures(d, features = character())

## S4 method for signature 'AnalysisData'
removeFeatures(d, features = character())

removeSamples(d, idx = "fileOrder", samples = c())

## S4 method for signature 'AnalysisData'
removeSamples(d, idx = "fileOrder", samples = c())
```

Arguments

d	S4 object of class AnalysisData
cls	info column to use for class information
classes	classes to remove
features	features to remove
idx	info column containing sample indexes
samples	sample indexes to remove

Value

An S4 object of class AnalysisData with samples, classes or features removed.

Methods

- removeClasses: Remove classes.
- removeFeatures: Remove features.
- removeSamples: Remove samples.

Examples

```
library(metaboData)
 d <- analysisData(abr1$neg[,200:300],abr1$fact)

 ## Remove classes
 d %>%
  removeClasses(cls = 'day',classes = 'H')

 ## Remove features
 d %>%
  removeFeatures(features = c('N200','N201'))

 ## Remove samples
 d %>%
  removeSamples(idx = 'injorder',samples = c(1,10))
```

---

roc *Receiver-operator characteristic (ROC) curves*

Description

ROC curves for out-of-bag random forest predictions.

Usage

```
roc(x)

## S4 method for signature 'RandomForest'
roc(x)

## S4 method for signature 'list'
roc(x)

## S4 method for signature 'Analysis'
roc(x)
```

Arguments

x	S4 object of class `RandomForest`, `Analysis` or a list

Value

A tibble containing the ROC curves.

Examples

```
library(metaboData)

x <- analysisData(abr1$neg[,200:300],abr1$fact) %>%
       occupancyMaximum(cls = 'day') %>%
       transformTICnorm()

rf <- randomForest(x,cls = 'day')

roc(rf)
```

---

rsd *Calculate feature relative standard deviations*

Description

Calculate relative standard deviation (RSD) percentage values for each feature per class for a given sample information column.

Usage

```
rsd(x, cls = "class")

## S4 method for signature 'AnalysisData'
rsd(x, cls = "class")
```

Arguments

x	S4 object of class AnalysisData
cls	sample information column to use for class structure

Value

A tibble containing the computed RSD values.

Examples

```
library(metaboData)

d <- analysisData(abr1$neg[,200:300],abr1$fact)

rsd(d,cls = 'day')
```

---

split *Split an* AnalysisData *object*

Description

Split an object of class AnalysisData into a list based a class grouping variable.

Usage

```
split(x, cls = "class")

## S4 method for signature 'AnalysisData'
split(x, cls = "class")
```

Arguments

x	S4 object of class AnalysisData
cls	sample information column to use for splitting

Value

A list of AnalysisData objects.

Examples

```
library(metaboData)

d <- analysisData(abr1$neg,abr1$fact)

## Split the data set based on the 'day' class information column
d <- split(d,cls = 'day')

print(d)
```

transformArcSine *Scaling, transformation and normalisation methods*

Description

Methods for data scaling, transformation and normalisation.

Usage

```
transformArcSine(d)

## S4 method for signature 'AnalysisData'
transformArcSine(d)

transformAuto(d)

## S4 method for signature 'AnalysisData'
transformAuto(d)

transformCenter(d)

## S4 method for signature 'AnalysisData'
transformCenter(d)

transformLevel(d)

## S4 method for signature 'AnalysisData'
transformLevel(d)

transformLn(d, add = 1)

## S4 method for signature 'AnalysisData'
transformLn(d, add = 1)

transformLog10(d, add = 1)

## S4 method for signature 'AnalysisData'
transformLog10(d, add = 1)

transformPareto(d)

## S4 method for signature 'AnalysisData'
transformPareto(d)

transformPercent(d)

## S4 method for signature 'AnalysisData'
```

```
transformPercent(d)

transformRange(d)

## S4 method for signature 'AnalysisData'
transformRange(d)

transformSQRT(d)

## S4 method for signature 'AnalysisData'
transformSQRT(d)

transformTICnorm(d, refactor = TRUE)

## S4 method for signature 'AnalysisData'
transformTICnorm(d, refactor = TRUE)

transformVast(d)

## S4 method for signature 'AnalysisData'
transformVast(d)
```

Arguments

d	S4 object of class AnalysisData
add	value to add prior to transformation
refactor	TRUE/FALSE. Re-factor the normalised intensity values to a range consistent with the raw values by multiplying by the median sample TIC.

Details

Prior to downstream analyses, metabolomics data often require transformation to fulfil the assumptions of a particular statistical/data mining technique. Before applying a transformation, it is important to consider the effects that the transformation will have on the data, as this can greatly effect the outcome of further downstream analyses. It is also important to consider at what stage in the pre-treatment routine a transformation is applied as this too could introduce artefacts into the data. The best practice is to apply a transformation as the last in a pre-treatment routine after all other steps have been taken. There are a wide range of transformation methods available that are commonly used for the analysis of metabolomics data.

Value

An S4 object of class AnalysisData containing the transformed data.

Methods

- transformArcSine: Arc-sine transformation.
- transformAuto: Auto scaling.

- `transformCenter`: Mean centring.
- `transformLevel`: Level scaling.
- `transformLn`: Natural logarithmic transformation.
- `transformLog10`: Logarithmic transformation.
- `transformPareto`: Pareto scaling.
- `transformPercent`: Scale as a percentage of the feature maximum intensity.
- `transformRange`: Range scaling. Also known as min-max scaling.
- `transformSQRT`: Square root transformation.
- `transformTICnorm`: Total ion count normalisation.
- `transformVast`: Vast scaling.

Examples

```
## Each of the following examples shows the application of the transformation and then
## a Linear Discriminant Analysis is plotted to show it's effect on the data structure.

## Initial example data preparation
library(metaboData)

d <- analysisData(abr1$neg[,200:300],abr1$fact) %>%
 occupancyMaximum(occupancy = 2/3)

d %>%
 plotLDA(cls = 'day')


## Arc-sine transformation
d %>%
 transformArcSine() %>%
 plotLDA(cls = 'day')

## Auto scaling
d %>%
 transformAuto() %>%
 plotLDA(cls = 'day')

## Mean centring
d %>%
 transformCenter()%>%
 plotLDA(cls = 'day')

## Level scaling
d %>%
 transformLevel() %>%
 plotLDA(cls = 'day')

## Natural logarithmic transformation
d %>%
 transformLn() %>%
```

```
 plotLDA(cls = 'day')

## Logarithmic transformation
d %>%
 transformLog10()%>%
 plotLDA(cls = 'day')

## Pareto scaling
d %>%
 transformPareto() %>%
 plotLDA(cls = 'day')

## Percentage scaling
d %>%
 transformPercent() %>%
 plotLDA(cls = 'day')

## Range scaling
d %>%
 transformRange() %>%
 plotLDA(cls = 'day')

## Square root scaling
d %>%
 transformSQRT() %>%
 plotLDA(cls = 'day')

## Total ion count nromalisation
d %>%
 transformTICnorm() %>%
 plotLDA(cls = 'day')

## Vast scaling
d %>%
 transformVast() %>%
 plotLDA(cls = 'day')
```

`ttest` *Welch's t-test*

Description

Welch's t-test

Usage

```
ttest(
  x,
  cls = "class",
  pAdjust = "bonferroni",
```

```
  comparisons = list(),
  returnModels = FALSE
)

## S4 method for signature 'AnalysisData'
ttest(
  x,
  cls = "class",
  pAdjust = "bonferroni",
  comparisons = list(),
  returnModels = FALSE
)
```

Arguments

x	S4 object of class AnalysisData
cls	vector of sample information column names to analyse
pAdjust	p value adjustment method
comparisons	named list of binary comparisons to analyse
returnModels	should models be returned

Value

An S4 object of class `Univariate`.

Examples

```
library(metaboData)

d <- analysisData(abr1$neg[,200:300],abr1$fact) %>%
 keepClasses(cls = 'day',classes = c('H','5'))

## Perform t-test
ttest_analysis <- ttest(d,cls = 'day')

## Extract significant features
explanatoryFeatures(ttest_analysis)
```

---

tune *Tune random forest parameters*

Description

Tune the `mtry` and `ntree` random forest parameters using a grid search approach.

Usage

```
tune(
  x,
  cls = "class",
 mtry_range = floor(seq(mtry(x, cls = cls) - mtry(x, cls = cls)/2, mtry(x, cls = cls) +
    mtry(x, cls = cls)/2, length.out = 4)),
  ntree_range = 1000,
  seed = 1234
)

## S4 method for signature 'AnalysisData'
tune(
  x,
  cls = "class",
 mtry_range = floor(seq(mtry(x, cls = cls) - mtry(x, cls = cls)/2, mtry(x, cls = cls) +
    mtry(x, cls = cls)/2, length.out = 4)),
  ntree_range = 1000,
  seed = 1234
)
```

Arguments

x	S4 object of class AnalysisData
cls	sample information column to use
mtry_range	numeric vector of mtry values to search
ntree_range	numeric vector of ntree values to search
seed	random number seed

Details

Parameter tuning is performed by grid search of all combinations of the mtry_range and ntree_range vectors provided. The optimal parameter values are selected using the out-of-bag error estimates of the margin metric for classification and the rmse (root-mean-square error) metric for regression.

Value

A list containing the optimal mtry and ntree parameters. This is suitable for use as the rf argument in method randomForest().

Examples

```
library(metaboData)

## Prepare some data
x <- analysisData(abr1$neg[,200:300],abr1$fact) %>%
  occupancyMaximum(cls = 'day') %>%
  transformTICnorm()
```

```
## Tune the `mtry` parameter for the `day` response
tune(x,cls = 'day')
```

`Univariate-class` *Univariate S4 class*

Description

An S4 class for univariate test models and results.

Slots

`type` univariate test type

`models` list of model objects

`results` tibble containing test results

Index