Package 'SIAMCAT'

October 16, 2018

Type Package

Title Statistical Inference of Associations between Microbial Communities And host phenoTypes

Version 1.0.0

Description Pipeline for Statistical Inference of Associations between Microbial Communities And host phenoTypes (SIAMCAT). A primary goal of analyzing microbiome data is to determine changes in community composition that are associated with environmental factors. In particular, linking human microbiome composition to host phenotypes such as diseases has become an area of intense research. For this, robust statistical modeling and biomarker extraction toolkits are crucially needed. SIAMCAT provides a full pipeline supporting data preprocessing, statistical association testing, statistical modeling (LASSO logistic regression) including tools for evaluation and interpretation of these models (such as cross validation, parameter selection, ROC analysis and diagnostic model plots).

Depends R (>= 3.5.0), mlr, phyloseq

Imports beanplot, glmnet, graphics, grDevices, grid, gridBase, gridExtra, LiblineaR, matrixStats, methods, ParamHelpers, pROC, PRROC, RColorBrewer, stats, utils

License GPL-3

LazyData true

RoxygenNote 6.0.1.9000

biocViews Metagenomics, Classification, Microbiome, Sequencing, Preprocessing, Clustering, FeatureExtraction, GeneticVariability, MultipleComparison, Regression

Suggests BiocStyle, optparse, testthat, knitr, rmarkdown

VignetteBuilder knitr

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

git_branch RELEASE_3_7

git_last_commit b5e27eb

git_last_commit_date 2018-04-30

Date/Publication 2018-10-15

Author Georg Zeller [aut] (<https://orcid.org/0000-0003-1429-7485>),
Konrad Zych [aut, cre] (<https://orcid.org/0000-0001-7426-0516>),
Jakob Wirbel [aut] (<https://orcid.org/0000-0002-4073-3562>),
Morgan Essex [ctb],
Nicolai Karcher [ctb],
Kersten Breuer [ctb]

Maintainer Konrad Zych <konrad.zych@embl.de>

R topics documented:

`SIAMCAT-package` *SIAMCAT: Statistical Inference of Associations between Microbial Communities And host phenoTypes*

Description

Pipeline for Statistical Inference of Associations between Microbial Communities And host phenoTypes (SIAMCAT). A primary goal of analyzing microbiome data is to determine changes in community composition that are associated with environmental factors. In particular, linking human microbiome composition to host phenotypes such as diseases has become an area of intense research. For this, robust statistical modeling and biomarker extraction toolkits are crucially needed. SIAMCAT provides a full pipeline supporting data preprocessing, statistical association testing, statistical modeling (LASSO logistic regression) including tools for evaluation and interpretation of these models (such as cross validation, parameter selection, ROC analysis and diagnostic model plots).

Details

SIAMCAT is a pipeline for Statistical Inference of Associations between Microbial Communities And host phenoTypes. A primary goal of analyzing microbiome data is to determine changes in community composition that are associated with environmental factors. In particular, linking human microbiome composition to host phenotypes such as diseases has become an area of intense research. For this, robust statistical modeling and biomarker extraction toolkits are crucially needed!

Author(s)

Maintainer: Konrad Zych <konrad.zych@embl.de> (0000-0001-7426-0516)

Authors:

- Georg Zeller <zeller@embl.de> (0000-0003-1429-7485)
- Jakob Wirbel <jakob.wirbel@embl.de> (0000-0002-4073-3562)

Other contributors:

- Morgan Essex <morgan.essex@embl.de> [contributor]
- Nicolai Karcher [contributor]
- Kersten Breuer [contributor]

accessSlot *Universal slot accessor function for siamcat-class.*

Description

This function is used internally by many accessors.

Usage

```
accessSlot(siamcat, slot)
```

Arguments

siamcat	an object of siamcat-class.
slot	A character string indicating the slot (not data class) of the component data type that is desired.

Value

Returns the component object specified by the argument `slot`. Returns NULL if slot does not exist.

Examples

```
#
data(siamcat_example)
accessSlot(siamcat_example, "label")
accessSlot(siamcat_example, "model_list")
```

add.meta.pred *Add metadata as predictors*

Description

This function adds metadata to the feature matrix to be later used as predictors

Usage

```
add.meta.pred(siamcat, pred.names = NULL, std.meta =
    TRUE, verbose = 1)
```

Arguments

siamcat	object of class siamcat-class
pred.names	vector of names of the variables within the metadata to be added to the feature matrix as predictors
std.meta	boolean, should added metadata features be standardized?, defaults to TRUE
verbose	control output: 0 for no output at all, 1 for only information about progress and success, 2 for normal level of information and 3 for full debug information, defaults to 1

Value

an object of class siamcat-class with metadata added to the features

Examples

```
data(siamcat_example)
# Add the Age of the patients as potential predictor
siamcat_age_added <- add.meta.pred(siamcat_example, pred.names=c('age'))

# Add Age, BMI, and Gender as potential predictors
# Additionally, prevent standardization of the added features
siamcat_meta_added <- add.meta.pred(siamcat_example, pred.names=c('age',
'bmi', 'gender'), std.meta=FALSE)
```

---

check.associations *Check and visualize associations between features and classes*

Description

This function calculates for each feature a pseudo-fold change (geometrical mean of the difference between quantiles) between the different classes found in labels.

Significance of the differences is computed for each feature using a Wilcoxon test followed by multiple hypothesis testing correction.

Additionally, the Area Under the Receiver Operating Characteristic Curve (AU-ROC) and a prevalence shift are computed for the features found to be associated with the two different classes at a user-specified significance level `alpha`.

Finally, the function produces a plot of the top `max.show` associated features, showing the distribution of the log10-transformed abundances for both classes, and user-selected panels for the effect (AU-ROC, Prevalence Shift, and Fold Change)

Usage

```
check.associations(siamcat,fn.plot,color.scheme = "RdYlBu",
    alpha =0.05,mult.corr = "fdr", sort.by = "fc",detect.lim = 1e-06,
    pr.cutoff = 1e-6, max.show = 50, plot.type = "quantile.box",
    panels = c("fc","auroc"),verbose = 1)
```

Arguments

`siamcat`	object of class siamcat-class
`fn.plot`	filename for the pdf-plot
`color.scheme`	valid R color scheme or vector of valid R colors (must be of the same length as the number of classes), defaults to `'RdYlBu'`
`alpha`	float, significance level, defaults to `0.05`
`mult.corr`	multiple hypothesis correction method, see `p.adjust`, defaults to `"fdr"`
`sort.by`	string, sort features by p-value (`"p.val"`), by fold change (`"fc"`) or by prevalence shift (`"pr.shift"`), defaults to `"fc"`
`detect.lim`	float, pseudocount to be added before log-transformation of the data, defaults to `1e-06`

pr.cutoff	float, cutoff for the prevalence computation, defaults to 1e-06
max.show	integer, how many associated features should be shown, defaults to 50
plot.type	string, specify how the abundance should be plotted, must be one of these: c("bean", "box", "quantile.box", "quantile.rect"), defaults to "quantile.box"
panels	vector, name of the panels to be plotted next to the log10- transformed abundances, possible entries are c("fc", "auroc","prevalence"), defaults to c("fc", "auroc")
verbose	control output: 0 for no output at all, 1 for only information about progress and success, 2 for normal level of information and 3 for full debug information, defaults to 1

Value

Does not return anything, but produces an association plot

Examples

```
# Example data
    data(siamcat_example)
# since the whole pipeline has been run in the example data, exchange the
# normalized features with the original features
    siamcat_example <- reset.features(siamcat_example)

# Simple example
    check.associations(siamcat_example, './assoc_plot.pdf')

# Plot associations as bean plot
    check.associations(siamcat_example, './assoc_plot_bean.pdf',
    plot.type='bean')

# Plot assocations as box plot
# Additionally, sort by p-value instead of by fold change
    check.associations(siamcat_example, './assoc_plot_fc.pdf',
    plot.type='box', sort.by='p.val')

# Custom colors
    check.associations(siamcat_example, './assoc_plot_blue_yellow.pdf',
    plot.type='box', color.scheme=c('cornflowerblue', '#ffc125'))
```

---

check.confounders *Check for potential confounders in the metadata*

Description

This function checks for associations between class labels and potential confounders (e.g. age, sex, or BMI) that are present in the metadata. Statistical testing is performed with Fisher's exact test or Wilcoxon test, while associations are visualized either as barplot or Q-Q plot, depending on the type of metadata.

Usage

```
check.confounders(siamcat, fn.plot, verbose = 1)
```

Arguments

siamcat	an object of class siamcat-class
fn.plot	string, filename for the pdf-plot
verbose	control output: 0 for no output at all, 1 for only information about progress and success, 2 for normal level of information and 3 for full debug information, defaults to 1

Value

Does not return anything, but produces a single plot for each metadata category.

Examples

```
    # Example data
    data(siamcat_example)
    # since the whole pipeline has been run in the example data, exchange the
    # normalized features with the original features
    siamcat_example <- reset.features(siamcat_example)

    # Simple working example
    check.confounders(siamcat_example, './conf_plot.pdf')

    # Additional information with verbose
    ## Not run:
    check.confounders(siamcat_example, './conf_plot.pdf',
    verbose=2)
## End(Not run)
```

---

create.data.split *Split a dataset into training and a test sets.*

Description

This function prepares the cross-validation by splitting the data into num.folds training and test folds for num.resample times.

Usage

```
create.data.split(siamcat, num.folds = 2, num.resample = 1,
    stratify = TRUE,inseparable = NULL, verbose = 1)
```

Arguments

siamcat	object of class siamcat-class
num.folds	number of cross-validation folds (needs to be >=2), defaults to 2
num.resample	resampling rounds (values <= 1 deactivate resampling), defaults to 1
stratify	boolean, should the splits be stratified so that an equal proportion of classes are present in each fold?, defaults to TRUE
inseparable	column name of metadata variable, defaults to NULL
verbose	control output: 0 for no output at all, 1 for only information about progress and success, 2 for normal level of information and 3 for full debug information, defaults to 1

Details

This function splits the labels within a siamcat-class object and prepares the internal cross-validation for the model training (see train.model).

The function saves the training and test instances for the different cross-validation folds within a list in the `data_split`-slot of the siamcat-class object, which is a list with four entries:

- `num.folds` the number of cross-validation folds
- `num.resample` the number of repetitions for the cross-validation
- `training.folds` a list containing the indices for the training instances
- `test.folds` a list containing the indices for the test instances

Value

object of class siamcat-class with the `data_split`-slot filled

Examples

```
data(siamcat_example)
# simple working example
siamcat_split <- create.data.split(siamcat_example, num.folds=10,
num.resample=5, stratify=TRUE)

## # example with a variable which is to be inseparable
## siamcat_split <- create.data.split(siamcat_example, num.folds=10,
##  num.resample=5, stratify=FALSE, inseparable='Gender')
```

---

data_split *Retrieve a data_split-class object from object.*

Description

Retrieve a data_split-class object from object.

Usage

```
data_split(siamcat)

## S4 method for signature 'ANY'
data_split(siamcat)

## S4 method for signature 'data_split'
data_split(siamcat)

## S4 method for signature 'list'
data_split(siamcat)
```

Arguments

`siamcat`	(Required). An instance of siamcat-class that contains a label or instance of data_split-class or a list.

Value

The data_split-class object or NULL.

Examples

```
data(siamcat_example)
data_split(siamcat_example)
```

---

data_split-class *The S4 class for storing data splits*

Description

The S4 class for storing data splits

Slots

`.Data` inherited from `list` class, contains a list with:

- `training.folds` a list - for each cv fold contains ids of samples used for training
- `test.folds` a list - for each cv fold contains ids of samples used for testing
- `num.resample` number of repetition rounds for cv
- `num.folds` number of folds for cv

---

data_split<- *Assign a new data_split object to* x

Description

Assign a new data_split object to x

Usage

```
data_split(x) <- value

## S4 replacement method for signature 'siamcat,data_split'
data_split(x) <- value
```

Arguments

x	an object of class siamcat-class
value	an object of class data_split-class

Value

none

Examples

```
data(siamcat_example)
data_split(siamcat_example) <- data_split(siamcat_example)
```

`evaluate.predictions` *Evaluate prediction results*

Description

This function takes the correct labels and predictions for all samples and evaluates the results using the

- Area Under the Receiver Operating Characteristic (ROC) Curve (AU-ROC)
- and the Precision-Recall Curve (PR)

as metric. Predictions can be supplied either for a single case or as matrix after resampling of the dataset.

Prediction results are usually produced with the function make.predictions.

Usage

```
evaluate.predictions(siamcat, verbose = 1)
```

Arguments

`siamcat`	object of class siamcat-class
`verbose`	control output: `0` for no output at all, `1` for only information about progress and success, `2` for normal level of information and `3` for full debug information, defaults to `1`

Details

This functions calculates for the predictions in the `pred_matrix` -slot of the siamcat-class-object several metrices. The Area Under the Receiver Operating Characteristic (ROC) Curve (AU-ROC) and the Precision-Recall Curve will be evaluated and the results will be saved in the `eval_data`-slot of the supplied siamcat-class-object. The `eval_data`-slot contains a list with several entries:

- `$roc.average` average ROC-curve across repeats or a single ROC-curve on complete dataset;
- `$auc.average` AUC value for the average ROC-curve;
- `$ev.list` list of `length(num.folds)`, containing for different decision thresholds the number of false positives, false negatives, true negatives, and true positives;
- `$pr.list` list of `length(num.folds)`, containing the positive predictive value (precision) and true positive rate (recall) values used to plot the PR curves.

For the case of repeated cross-validation, the function will additonally return

- `$roc.all` list of roc objects (see roc) for every repeat;
- `$aucspr` vector of AUC values for the PR curves for every repeat;
- `$auc.all` vector of AUC values for the ROC curves for every repeat.

Value

object of class siamcat-class with the slot `eval_data` filled

Examples

```
data(siamcat_example)
# simple working example
siamcat_evaluated <- evaluate.predictions(siamcat_example)
```

`eval_data` *Retrieve eval_data from object.*

Description

Retrieve eval_data from object.

Usage

```
eval_data(siamcat)

## S4 method for signature 'ANY'
eval_data(siamcat)

## S4 method for signature 'list'
eval_data(siamcat)
```

Arguments

`siamcat`	(Required). An instance of siamcat-class that contains a eval_data..

Value

The eval_data list or NULL.

Examples

```
data(siamcat_example)
eval_data(siamcat_example)
```

`eval_data-class` *The S4 class for storing evaluation data.*

Description

The S4 class for storing evaluation data.

Slots

.Data inherited from list class, contains a list with:

- $roc.average average ROC-curve across repeats or a single ROC-curve on complete dataset;
- $auc.average AUC value for the average ROC-curve;
- $ev.list list of length(num.folds), containing for different decision thresholds the number of false positives, false negatives, true negatives, and true positives;
- $pr.list list of length(num.folds), containing the positive predictive value (precision) and true positive rate (recall) values used to plot the PR curves;

. If prediction had more than one column, i.e. if the models has been trained with several repeats, the function will additonally return

- $roc.all list of roc objects (see roc) for every repeat;
- $aucspr vector of AUC values for the PR curves for every repeat;
- $auc.all vector of AUC values for the ROC curves for every repeat

---

eval_data<- *Assign a new eval_data object to* x

Description

Assign a new eval_data object to x

Usage

```
eval_data(x) <- value

## S4 replacement method for signature 'siamcat,list'
eval_data(x) <- value
```

Arguments

x	an object of class siamcat-class
value	an eval_data list

Value

none

Examples

```
data(siamcat_example)
eval_data(siamcat_example) <- eval_data(siamcat_example)
```

features *Retrieve a otu_table-class object from object.*

Description

Retrieve a otu_table-class object from object.

Usage

```
features(siamcat)

## S4 method for signature 'ANY'
features(siamcat)

## S4 method for signature 'otu_table'
features(siamcat)
```

Arguments

siamcat	(Required). An instance of siamcat-class that contains a label or instance of otu_table-class .

Value

The otu_table-class object or NULL.

Examples

```
data(siamcat_example)
features(siamcat_example)
```

features<- *Assign a new otu_table object to* x *features slot*

Description

Assign a new otu_table object to x features slot

Usage

```
features(x) <- value

## S4 replacement method for signature 'siamcat,otu_table'
features(x) <- value
```

Arguments

x	an object of class siamcat-class
value	an object of class otu_table-class

Value

none

Examples

```
data(siamcat_example)
features(siamcat_example) <- features(siamcat_example)
```

`filter.features` *Perform unsupervised feature filtering.*

Description

This function performs unsupervised feature filtering. Features can be filtered based on abundance or prevalence. Additionally, unmapped reads may be removed.

Usage

```
filter.features(siamcat, filter.method = "abundance",
    cutoff = 0.001, recomp.prop = FALSE, rm.unmapped = TRUE, verbose = 1)
```

Arguments

`siamcat`	an object of class siamcat-class
`filter.method`	method used for filtering the features, can be one of these: `c('abundance', 'cum.abundance', 'pr` defaults to `'abundance'`
`cutoff`	float, abundace or prevalence cutoff, default to `0.001`
`recomp.prop`	boolean, should relative abundances be recomputed?, defaults to `FALSE`
`rm.unmapped`	boolean, should unmapped reads be discarded?, defaults to `TRUE`
`verbose`	control output: `0` for no output at all, `1` for only information about progress and success, `2` for normal level of information and `3` for full debug information, defaults to `1`

Details

This function filters the features in a siamcat-class object in a unsupervised manner.

The different filter methods work in the following way:

- `'abundace'` remove features whose abundance is never above the threshold value in any of the samples
- `'cum.abundance'` remove features with very low abundance in all samples i.e. ones that are never among the most abundant entities that collectively make up (1-cutoff) of the reads in any sample
- `'prevalence'` remove features with low prevalence across samples i.e. ones that are 0 (undetected) in more than (1-cutoff) proportion of samples.

Value

siamcat an object of class siamcat-class

Examples

```
    # Example dataset
    data(siamcat_example)
    # since the whole pipeline has been run in the example data, the feature
    # were filtered already.
    siamcat_example <- reset.features(siamcat_example)

# Simple examples
siamcat_filtered <- filter.features(siamcat_example,
    filter.method='abundance',
    cutoff=1e-03)
```

`filter.label` *Filter samples from* `siamcat@label`

Description

This functions filters `siamcat@label`.

Usage

```
filter.label(siamcat, ids, verbose = 1)
```

Arguments

`siamcat`	an object of class siamcat-class
`ids`	names of samples to be left in the `siamcat@label`
`verbose`	control output: `0` for no output at all, `1` for more information about progress and success, defaults to `1`

Value

siamcat an object of class siamcat-class

Examples

```
    data(siamcat_example)
    # simple working example
    siamcat_filtered <- filter.label(siamcat_example, ids=c(1:10))
```

get.features.matrix *get.features.matrix*

Description

Function to access features in siamcat@phylose@otu_table

Usage

```
get.features.matrix(siamcat)
```

Arguments

siamcat	an object of class siamcat-classt

Details

Access features in siamcat@phylose@otu_table as matrix

Value

Features as a matrix

Examples

```
data(siamcat_example)
feat <- get.features.matrix(siamcat_example)
```

get.orig_feat.matrix *get.orig_feat.matrix*

Description

Function to access original features in siamcat@orig_feat

Usage

```
get.orig_feat.matrix(siamcat)
```

Arguments

siamcat	an object of class siamcat-classt

Details

Access original features in siamcat@orig_feat as matrix

Value

Original features as a matrix

Examples

```
data(siamcat_example)
orig_feat <- get.orig_feat.matrix(siamcat_example)
```

`label` *Retrieve a label-class object from object.*

Description

Retrieve a label-class object from object.

Usage

```
label(siamcat)

## S4 method for signature 'ANY'
label(siamcat)

## S4 method for signature 'label'
label(siamcat)

## S4 method for signature 'list'
label(siamcat)

## S4 method for signature 'otu_table'
orig_feat(siamcat)
```

Arguments

`siamcat`	(Required). An instance of siamcat-class that contains a label or instance of label-class or a list.

Value

The label-class object or NULL.

Examples

```
data(siamcat_example)
label(siamcat_example)
```

label-class *The S4 class for storing label info.*

Description

The S4 class for storing label info.

Slots

`.Data` inherited from `list` class, contains a list with:

- `label` numeric vector, specifying to which category samples belong, usualy made of 1s and -1s
- `header` contains information from the header of the label file
- `info` list with additional informations about the dataset
- `positive.lab` specifies which of two numbers in label is a positive label
- `negative.lab` specifies which of two numbers in label is a negative label
- `n.idx` numeric vector - on which positions in the label there are samples with negative label
- `p.idx` numeric vector - on which positions in the label there are samples with positive label
- `n.lab` character string with a name for the negative label (e.g. 'healthy')
- `p.lab` character string with a name for the positive label (e.g. 'cancer')

label<- *Assign a new label object to* `x`

Description

Assign a new label object to `x`

Usage

```
label(x) <- value

## S4 replacement method for signature 'siamcat,label'
label(x) <- value
```

Arguments

`x`	an object of class siamcat-class
`value`	an object of class label-class

Value

none

Examples

```
data(siamcat_example)
label(siamcat_example) <- label(siamcat_example)
```

`make.predictions` *Make predictions on a test set*

Description

This function takes a siamcat-class-object containing a model trained by train.model and performs predictions on a given test-set.

Usage

```
make.predictions(siamcat, siamcat.holdout = NULL,
    normalize.holdout = TRUE, verbose = 1)
```

Arguments

`siamcat`	object of class siamcat-class
`siamcat.holdout`	optional, object of class siamcat-class on which to make predictions, defaults to `NULL`
`normalize.holdout`	boolean, should the holdout features be normalized with a frozen normalization (see normalize.features) using the normalization parameters in `siamcat`?, defaults to `TRUE`
`verbose`	control output: `0` for no output at all, `1` for only information about progress and success, `2` for normal level of information and `3` for full debug information, defaults to `1`

Details

This functions uses the model in the `model_list`-slot of the `siamcat` object to make predictions on a given test set. The test set can either consist of the test instances in the cross- validation, saved in the `data_split`-slot of the same `siamcat` object, or a completely external feature set, given in the form of another `siamcat` object (`siamcat.holdout`).

Value

object of class siamcat-class with the slot `pred_matrix` filled or a matrix containing the predictions for the holdout set

Examples

```
    data(siamcat_example)
    # Simple example
    siamcat.pred <- make.predictions(siamcat_example)

    # Predictions on a holdout-set
    ## Not run: pred.mat <- make.predictions(siamcat.trained, siamcat.holdout,
    normalize.holdout=TRUE)
## End(Not run)
```

meta *Retrieve a sample_data-class object from object.*

Description

Retrieve a sample_data-class object from object.

Usage

```
meta(siamcat)

## S4 method for signature 'ANY'
meta(siamcat)

## S4 method for signature 'sample_data'
meta(siamcat)
```

Arguments

siamcat	(Required). An instance of siamcat-class that contains a label or instance of sample_data-class.

Value

The sample_data-class object or NULL.

Examples

```
data(siamcat_example)
meta(siamcat_example)
```

meta<- *Assign a new sam_data object to* x

Description

Assign a new sam_data object to x

Usage

```
meta(x) <- value

## S4 replacement method for signature 'siamcat,sample_data'
meta(x) <- value
```

Arguments

x	an object of class siamcat-class
value	an object of class sample_data-class

Value

none

Examples

```
data(siamcat_example)
meta(siamcat_example) <- meta(siamcat_example)
```

`model.evaluation.plot` *Model Evaluation Plot*

Description

Produces two plots for model evaluation. The first plot shows the Receiver Operating Characteristic (ROC)-curves, the other the Precision-recall (PR)-curves for the different cross-validation repetitions.

Usage

```
model.evaluation.plot(siamcat, fn.plot, verbose = 1)
```

Arguments

`siamcat`	object of class siamcat-class
`fn.plot`	string, filename for the pdf-plot
`verbose`	control output: `0` for no output at all, `1` for only information about progress and success, `2` for normal level of information and `3` for full debug information, defaults to `1`

Value

Does not return anything, but produces the model evaluation plot.

Examples

```
data(siamcat_example)
# simple working example
model.evaluation.plot(siamcat_example, fn.plot='./eval,pdf')
```

`model.interpretation.plot` *Model Interpretation Plot*

Description

Produces a plot for model interpretation, displaying feature weights, robustness of feature weights, and features scores across patients.

Usage

```
model.interpretation.plot(siamcat, fn.plot, color.scheme = "BrBG",
consens.thres = 0.5,heatmap.type = c("zscore", "fc"),
norm.models = FALSE, limits = c(-3, 3), detect.lim = 1e-06,
max.show = 50, verbose = 1)
```

Arguments

`siamcat`	object of class siamcat-class
`fn.plot`	string, filename for the pdf-plot
`color.scheme`	color scheme for the heatmap, defaults to `'BrBG'`
`consens.thres`	minimal ratio of models incorporating a feature in order to include it into the heatmap, defaults to `0.5` Note that for `'randomForest'` models, this cutoff specifies the minimum median Gini coefficient for a feature to be included and should therefore be much lower, e.g. `0.01`
`heatmap.type`	type of the heatmap, can be either `'fc'` or `'zscore'`, defaults to `'zscore'`
`norm.models`	boolean, should the feature weights be normalized across models?, defaults to `FALSE`
`limits`	vector, cutoff for extreme values in the heatmap, defaults to `c(-3, 3)`
`detect.lim`	float, pseudocount to be added before log-transformation of features, defaults to `1e-06`
`max.show`	integer, maximum number of features to be shown in the model interpretation plot, defaults to 50
`verbose`	control output: `0` for no output at all, `1` for only information about progress and success, `2` for normal level of information and `3` for full debug information, defaults to `1`

Details

Produces a plot consisting of

- a barplot showing the feature weights and their robustness (i.e. in what proportion of models have they been incorporated)
- a heatmap showing the z-scores of the metagenomic features across patients
- another heatmap displaying the metadata categories (if applicable)
- a boxplot displaying the poportion of weight per model that is actually shown for the features that are incorporated into more than `consens.thres` percent of the models.

Value

Does not return anything, but produces the model interpretion plot.

Examples

```
data(siamcat_example)
# simple working example
model.interpretation.plot(siamcat_example, fn.plot='./interpretion,pdf',
heatmap.type='zscore')
```

`models` *Retrieve list of models from object.*

Description

Retrieve list of models from object.

Usage

```
models(siamcat)

## S4 method for signature 'ANY'
models(siamcat)
```

Arguments

`siamcat`	(Required). An instance of siamcat-class that contains a model_list or instance of model_list-class.

Value

The list of models or NULL.

Examples

```
data(siamcat_example)
models(siamcat_example)
```

model_list *Retrieve model_list-class from object.*

Description

Retrieve model_list-class from object.

Usage

```
model_list(siamcat)

## S4 method for signature 'ANY'
model_list(siamcat)

## S4 method for signature 'model_list'
model_list(siamcat)

## S4 method for signature 'model_list'
models(siamcat)

## S4 method for signature 'model_list'
model_type(siamcat)
```

Arguments

siamcat	(Required). An instance of siamcat-class that contains a model_list or instance of model_list-class.

Value

The model_list-class object or NULL.

Examples

```
data(siamcat_example)
model_list(siamcat_example)
```

model_list-class *The S4 class for storing models.*

Description

The S4 class for storing models.

Slots

`models` a list with models obtained from train.model

`model.type` name of the method used by train.model

model_list<- *Assign a new model_list object to* x

Description

Assign a new model_list object to x

Usage

```
model_list(x) <- value

## S4 replacement method for signature 'siamcat,model_list'
model_list(x) <- value
```

Arguments

x	an object of class siamcat-class
value	an object of class model_list-class

Value

none

Examples

```
data(siamcat_example)
model_list(siamcat_example) <- model_list(siamcat_example)
```

model_type *Retrieve model_type from object.*

Description

Retrieve model_type from object.

Usage

```
model_type(siamcat)

## S4 method for signature 'ANY'
model_type(siamcat)
```

Arguments

siamcat	(Required). An instance of siamcat-class that contains a model_list or instance of model_list-class.

Value

The string describing type of model used or NULL.

Examples

```
data(siamcat_example)
model_type(siamcat_example)
```

`normalize.features` *Perform feature normalization*

Description

This function performs feature normalization according to user- specified parameters.

Usage

```
normalize.features(siamcat,
norm.method = c("rank.unit", "rank.std", "log.std", "log.unit", "log.clr"),
norm.param = list(log.n0 = 1e-06, sd.min.q = 0.1, n.p = 2, norm.margin = 1),
verbose = 1)
```

Arguments

`siamcat`	an object of class siamcat-class
`norm.method`	string, normalization method, can be one of these: 'c('rank.unit', 'rank.std', 'log.std', 'lo
`norm.param`	list, specifying the parameters of the different normalization methods, see details for more information
`verbose`	control output: `0` for no output at all, `1` for only information about progress and success, `2` for normal level of information and `3` for full debug information, defaults to `1`

Details

There are five different normalization methods available:

- `'rank.unit'` converts features to ranks and normalizes each column (=sample) by the square root of the sum of ranks
- `'rank.std'` converts features to ranks and applies z-score standardization
- `'log.clr'` centered log-ratio transformation (with the addition of pseudocounts)
- `'log.std'` log-transforms features (after addition of pseudocounts) and applies z-score standardization
- `'log.unit'` log-transforms features (after addition of pseudocounts) and normalizes by features or samples with different norms

The list entries in `'norm.param'` specify the normalzation parameters, which are dependant on the normalization method of choice:

- `'rank.unit'` does not require any other parameters
- `'rank.std'` requires `sd.min.q`, quantile of the distribution of standard deviations of all features that will be added to the denominator during standardization in order to avoid underestimation of the standard deviation, defaults to 0.1
- `'clr'` requires `log.n0`, which is the pseudocount to be added before log-transformation, defaults to `NULL` leading to the estimation of `log.n0` from the data

- 'log.std' requires both log.n0 and sd.min.q, using the same default values
- 'log.unit' requires next to log.n0 also the parameters n.p and norm.margin. n.p specifies the vector norm to be used, can be either 1 for x/sum(x) or 2 for x/sqrt(sum(x^2)). The parameter norm.margin specifies the margin over which to normalize, similarly to the apply-syntax: Allowed values are 1 for normalization over features, 2 over samples, and 3 for normalization by the global maximum.

The function additionally allows to perform a frozen normalization on a different dataset. After normalizing the first dataset, the output list $par contains all parameters of the normalization. Supplying this list together with a new dataset will normalize the second dataset in a comparable way to the first dataset (e.g. by using the same mean for the features for z-score standardization)

Value

an object of class siamcat-class with normalized features

Examples

```
# Example data
data(siamcat_example)
# since the whole pipeline has been run in the example data, exchange the
# normalized features with the original features
siamcat_example <- reset.features(siamcat_example)

# Simple example
siamcat_norm <- normalize.features(siamcat_example,
norm.method='rank.unit')

# log.unit example
siamcat_norm <- normalize.features(siamcat_example,
norm.method='log.unit', norm.param=list(log.n0=1e-05, n.p=1,
norm.margin=1))

# log.std example
siamcat_norm <- normalize.features(siamcat_example,
norm.method='log.std', norm.param=list(log.n0=1e-05, sd.min.q=.1))
```

---

norm_param *Retrieve norm_param from object.*

Description

Retrieve norm_param from object.

Usage

```
norm_param(siamcat)

## S4 method for signature 'ANY'
norm_param(siamcat)
```

Arguments

siamcat	(Required). An instance of siamcat-class that contains a norm_param

Value

The norm_param list or NULL.

Examples

```
data(siamcat_example)
norm_param(siamcat_example)
```

---

norm_param<- *Assign a new norm_param object to* x

Description

Assign a new norm_param object to x

Usage

```
norm_param(x) <- value

## S4 replacement method for signature 'siamcat,list'
norm_param(x) <- value
```

Arguments

x	an object of class siamcat-class
value	an norm_param list

Value

none

Examples

```
data(siamcat_example)
norm_param(siamcat_example) <- norm_param(siamcat_example)
```

orig_feat *Retrieve a otu_table-class object from orig_feat slot.*

Description

Retrieve a otu_table-class object from orig_feat slot.

Usage

```
orig_feat(siamcat)

## S4 method for signature 'ANY'
orig_feat(siamcat)

## S4 method for signature 'orig_feat'
orig_feat(siamcat)
```

Arguments

siamcat	(Required). An instance of siamcat-class that contains a label or instance of otu_table-class.

Value

The otu_table-class object or NULL.

Examples

```
data(siamcat_example)
data_split(siamcat_example)
```

orig_feat-class *The S4 class for storing original features info.*

Description

The S4 class for storing original features info.

Slots

`taxa_are_rows` A single logical specifying the orientation of the abundance table

`.Data` inherited from `matrix` class, contains a matrix with predictions made by make.predictions function

orig_feat<- *Assign a new otu_table object to* x *orig_feat slot*

Description

Assign a new otu_table object to x orig_feat slot

Usage

```
orig_feat(x) <- value

## S4 replacement method for signature 'siamcat,orig_feat'
orig_feat(x) <- value

## S4 replacement method for signature 'siamcat,otu_table'
orig_feat(x) <- value
```

Arguments

x	an object of class siamcat-class
value	an object of class otu_table-class

Value

none

Examples

```
data(siamcat_example)
orig_feat(siamcat_example) <- orig_feat(siamcat_example)
```

physeq *Retrieve a phyloseq-class object from object.*

Description

Retrieve a phyloseq-class object from object.

Usage

```
physeq(siamcat)

## S4 method for signature 'ANY'
physeq(siamcat)

## S4 method for signature 'phyloseq'
physeq(siamcat)
```

Arguments

siamcat	(Required). An instance of siamcat-class that contains a label or instance of phyloseq-class.

Value

The phyloseq-class object or NULL.

Examples

```
data(siamcat_example)
physeq(siamcat_example)
```

---

physeq<- *Assign a new phyloseq object to* x

Description

Assign a new phyloseq object to x

Usage

```
physeq(x) <- value

## S4 replacement method for signature 'siamcat,phyloseq'
physeq(x) <- value

## S4 replacement method for signature 'siamcat,otu_table'
physeq(x) <- value
```

Arguments

x	an object of class siamcat-class
value	an object of class phyloseq-class

Value

none

Examples

```
data(siamcat_example)
physeq(siamcat_example) <- physeq(siamcat_example)
```

pred_matrix *Retrieve pred_matrix from object.*

Description

Retrieve pred_matrix from object.

Usage

```
pred_matrix(siamcat)

## S4 method for signature 'ANY'
pred_matrix(siamcat)

## S4 method for signature 'matrix'
pred_matrix(siamcat)
```

Arguments

siamcat	(Required). An instance of siamcat-class that contains a pred_matrix

Value

The pred_matrix matrix or NULL.

Examples

```
data(siamcat_example)
pred_matrix(siamcat_example)
```

pred_matrix-class *The S4 class for storing label info.*

Description

The S4 class for storing label info.

Slots

.Data inherited from matrix class, contains a matrix with predictions made by make.predictions function

pred_matrix<- *Assign a new pred_matrix object to* x

Description

Assign a new pred_matrix object to x

Usage

```
pred_matrix(x) <- value

## S4 replacement method for signature 'siamcat,matrix'
pred_matrix(x) <- value
```

Arguments

x	an object of class siamcat-class
value	an pred_matrix matrix

Value

none

Examples

```
data(siamcat_example)
pred_matrix(siamcat_example) <- pred_matrix(siamcat_example)
```

read.features *Read feature file*

Description

This file reads in the tsv file with features and converts it into a matrix.

The file should be oragnized as follows: features (in rows) x samples (in columns).

First row should contain sample labels (consistent with label data), while the first column should contain feature labels (e.g. taxonomic identifiers). The remaining entries are expected to be real values >= 0 that quantify the abundance of each feature in each sample.

Usage

```
read.features(fn.in.feat, verbose = 0)
```

Arguments

fn.in.feat	name of the tsv file containing features
verbose	control output: 0 for no output at all, 1 for information about progress and time, defaults to 0

Value

otu_table containing features from the file

Examples

```
# run with example data
fn.feat <- system.file('extdata',
'feat_crc_study-pop-I_N141_tax_profile_mocat_bn_specI_clusters.tsv',
package = 'SIAMCAT')

features <- read.features(fn.feat)
```

---

read.labels *Read labels file*

Description

This file reads in the tsv file with labels and converts it into a label object.

First row is expected to be #BINARY:1=[label for cases]; -1=[label for controls]. Second row should contain the sample identifiers as tab-separated list (consistent with feature and metadata).

Third row is expected to contain the actual class labels (tab-separated): 1 for each case and -1 for each control.

Note: Labels can take other numeric values (but not characters or strings); importantly, the label for cases has to be greater than the one for controls

Usage

```
read.labels(fn.in.label)
```

Arguments

fn.in.label	name of the tsv file containing labels

Value

label object containing several entries:

- $label named vector containing the numerical labels from the file;
- $header first row of the label file;
- $info information about the type of label (e.g. BINARY);
- $positive.lab numerical label for controls, e.g. -1;
- $negative.lab numerical label for cases, e.g. 1;
- $n.idx logical vector of labels (TRUE for controls, FALSE otherwise);
- $n.lab label for controls, e.g. healthy;
- $p.idx logical vector of labels (TRUE for cases, FALSE otherwise);
- $p.lab label for cases, e.g. cancer

Examples

```
    # run with example data
fn.label <- system.file('extdata',
'label_crc_study-pop-I_N141_tax_profile_mocat_bn_specI_clusters.tsv',
    package = 'SIAMCAT')

labels <- read.labels(fn.label)
```

`read.meta` *Read metadata file*

Description

This file reads in the tsv file with numerical metadata and converts it into a matrix.

The file should be organized as follows: samples (in rows) x metadata (in columns). Metadata needs to be converted to numerical values by the user.

Metadata may be optional for the SIAMCAT workflow, but are necessary for heatmap displays, see model.interpretation.plot

Usage

```
read.meta(fn.in.meta)
```

Arguments

`fn.in.meta`	name of the tsv file containing metadata

Value

`sample_data` object

Examples

```
    # run with example data
fn.meta  <- system.file('extdata',
'num_metadata_crc_study-pop-I_N141_tax_profile_mocat_bn_specI_clusters.tsv',
package = 'SIAMCAT')

meta_data <- read.meta(fn.meta)
```

reset.features *reset.features*

Description

Function reset features in siamcat@phylose@otu_table to those in siamcat@orig_feat in an object of class siamcat-class

Usage

```
reset.features(siamcat)
```

Arguments

siamcat	an object of class siamcat-classt

Details

Reset features in siamcat@phylose@otu_table to those in siamcat@orig_feat

Value

A new siamcat-class object

Examples

```
data(siamcat_example)
siamcat_example <- reset.features(siamcat_example)
```

select.samples *Select samples based on metadata*

Description

This functions selects labels and metadata based on a specific column in the metadata. Provided with a column-name in the metadata and a range or a set of allowed values, the function will filter the siamcat-class object accordingly.

Usage

```
select.samples(siamcat, filter, allowed.set = NULL,
allowed.range = NULL, verbose = 1)
```

Arguments

siamcat	an object of class siamcat-class
filter	string, name of the meta variable on which the selection should be done
allowed.set	a vector of allowed values
allowed.range	a range of allowed values
verbose	control output: 0 for no output at all, 1 for only information about progress and success, 2 for normal level of information and 3 for full debug information, defaults to 1

Value

an object of class siamcat-class with labels and metadata filtered in order to contain only allowed values

Examples

```
    data(siamcat_example)
    # Select all samples that fall into an Age-range between 20 and 80 years
    siamcat_selected <- select.samples(siamcat_example, 'age',
    allowed.range=c(20, 80))

    # Select all samples for which information about the gender is given
    # Provide additional information with verbose
    ## Not run: siamcat_selected <- select.samples(siamcat_example, 'gender',
    allowed.set=c(1, 2), verbose=2)
## End(Not run)
```

`siamcat` *siamcat*

Description

Function to construct an object of class siamcat-class

Usage

```
siamcat(...)
```

Arguments

`...`	list of arguments needed in order to construct a SIAMCAT object

Details

Build siamcat-class objects from their components.

Value

A new siamcat-class object

Examples

```
# example with package data
fn.in.feat    <- system.file('extdata',
    'feat_crc_study-pop-I_N141_tax_profile_mocat_bn_specI_clusters.tsv',
    package = 'SIAMCAT')
fn.in.label <- system.file('extdata',
    'label_crc_study-pop-I_N141_tax_profile_mocat_bn_specI_clusters.tsv',
    package = 'SIAMCAT')
fn.in.meta    <- system.file('extdata',
'num_metadata_crc_study-pop-I_N141_tax_profile_mocat_bn_specI_clusters.tsv',
    package = 'SIAMCAT')
```

```
feat    <- read.features(fn.in.feat)
label <- read.labels(fn.in.label)
meta    <- read.meta(fn.in.meta)
siamcat <- siamcat(feat, label, meta)
```

`siamcat-class` *The S4 class for storing taxa-abundance information and models.*

Description

The S4 class for storing taxa-abundance information and models.

Slots

`phyloseq` object of class phyloseq-class

`label` an object of class label-class

`orig_feat` an object of class otu_table-class

`data_split` an object of class data_split-class

`norm_param` a list of normalzation parameters, see normalize.features for more details

`model_list` an object of class model_list-class

`eval_data` an object of class eval_data-class

`pred_matrix` an object of class pred_matrix-class

`siamcat_example` *Documentation for the example siamcat object in the data folder*

Description

Reduced version of the CRC dataset in inst/extdata, containing 100 features (15 associated features at 5% FDR in the original dataset and 85 random other features) and 141 samples, saved after the complete SIAMCAT pipelinehas been run. Therefore, contains entries in every siamcat-object slot, e.g, `eval_data` or `data_split`. Mainly used for running the examples in the function documentation

`train.model` *Model training*

Description

This function trains the a machine learning model on the training data

Usage

```
train.model(siamcat,
method = c("lasso","enet","ridge","lasso_ll", "ridge_ll", "randomForest"),
stratify = TRUE, modsel.crit = list("auc"), min.nonzero.coeff = 1,
param.set = NULL, verbose = 1)
```

Arguments

`siamcat`	object of class siamcat-class
`method`	string, specifies the type of model to be trained, may be one of these: `c('lasso', 'enet', 'ridge',`
`stratify`	boolean, should the folds in the internal cross-validation be stratified?, defaults to `TRUE`
`modsel.crit`	list, specifies the model selection criterion during internal cross-validation, may contain these: `c('auc', 'f1','acc', 'pr')`, defaults to `list('auc')`
`min.nonzero.coeff`	integer number of minimum nonzero coefficients that should be present in the model (only for `'lasso'`, `'ridge'`, and `'enet'`, defaults to `1`
`param.set`	a list of extra parameters for mlr run, may contain: • `cost` - for lasso_ll and ridge_ll • `alpha` for enet • `ntree` and `mtry` for RandomForrest. Defaults to `NULL`
`verbose`	control output: `0` for no output at all, `1` for only information about progress and success, `2` for normal level of information and `3` for full debug information, defaults to `1`

Details

This functions performs the training of the machine learning model and functions as an interface to the `mlr`-package.

The function expects a siamcat-class-object with a prepared cross-validation (see create.data.split) in the `data_split`-slot of the object. It then trains a model for each fold of the datasplit.

For the machine learning methods that require additional hyperparameters (e.g. `lasso_ll`), the optimal hyperparameters are tuned with the function tuneParams within the `mlr`-package.

The methods `'lasso'`, `'enet'`, and `'ridge'` are implemented as mlr-taks using the `'classif.cvglmnet'` Learner, `'lasso_ll'` and `'ridge_ll'` use the `'classif.LiblineaRL1LogReg'` and the `'classif.LiblineaRL2LogReg` Learners respectively. The `'randomForest'` method is implemented via the `'classif.randomForest'` Learner.

Value

object of class siamcat-class with added `model_list`

Examples

```
data(siamcat_example)
# simple working example
siamcat_validated <- train.model(siamcat_example, method='lasso')
```

`validate.data` *Validate samples in labels, features, and metadata*

Description

This function checks if labels are available for all samples in features. Additionally validates metadata, if available.

Usage

```
validate.data(siamcat, verbose = 1)
```

Arguments

`siamcat`	an object of class siamcat-class
`verbose`	control output: `0` for no output at all, `1` for only information about progress and success, `2` for normal level of information and `3` for full debug information, defaults to `1`

Details

This function validates the data by checking that labels are available for all samples in the feature matrix. Furthermore, the number of samples per class is checked to ensure a minimum number. If metadata is available, the overlap between labels and metadata is checked as well.

Value

an object of class siamcat-class with validated data

Examples

```
data(siamcat_example)
# simple working example
siamcat_validated <- validate.data(siamcat_example)
```

Index