Package ‘SIAMCAT’

April 16, 2019

Type Package

Title Statistical Inference of Associations between Microbial Communities And host phenoTypes

Version 1.2.1

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, scales, stats, stringr, utils, infotheo, corrplot

License GPL-3

LazyData true

RoxygenNote 6.1.0

biocViews ImmunoOncology, 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_8

git_last_commit 4f0a7ca

git_last_commit_date 2019-01-04

Date/Publication 2019-04-15

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

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

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, verbose=1)
```

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.
verbose	If the slot is empty, should a message be printed? values can be either 0 (no output) or 1 (print message)

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,
    std.meta = TRUE,
    feature.type='normalized',
    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`
`feature.type`	On which type of features should the function work? Can be either "original", "filtered", or "normalized". Please only change this paramter if you know what you are doing!
`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)
```

associations *Retrieve associations from object.*

Description

Retrieve associations from object.

Usage

```
associations(siamcat, verbose=1)

## S4 method for signature 'ANY'
associations(siamcat, verbose = 1)
```

Arguments

`siamcat`	(Required). An instance of siamcat-class that contains an object in the associations slot
`verbose`	If the slot is empty, should a message be printed? values can be either 0 (no output) or 1 (print message)

Value

The results of the association testing or NULL.

Examples

```
    data(siamcat_example)
    associations(siamcat_example)
```

associations-class *The S4 class for storing the results of the association testing*

Description

The S4 class for storing the results of the association testing

Slots

`assoc.results` a data.frame containing the results of the association testing

`assoc.param` a list containing the parameters for the association testing

`associations<-` *Assign a new assocications object to* `x`

Description

Assign a new assocications object to `x`

Usage

```
associations(x) <- value

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

Arguments

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

Value

none

Examples

```
data(siamcat_example)
associations(siamcat_example) <- new("associations",
    assoc.results=associations(siamcat_example),
    assoc.param=assoc_param(siamcat_example))
```

`assoc_param` *Retrieve parameters of association testing from object.*

Description

Retrieve parameters of association testing from object.

Usage

```
assoc_param(siamcat, verbose=1)

## S4 method for signature 'ANY'
assoc_param(siamcat, verbose = 1)
```

Arguments

`siamcat`	(Required). An instance of siamcat-class that contains an object in the associations slot
`verbose`	If the slot is empty, should a message be printed? values can be either 0 (no output) or 1 (print message)

Value

The parameters of the assocation testing or NULL

Examples

```
data(siamcat_example)
assoc_param(siamcat_example)
```

---

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=NULL, 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"), prompt = TRUE,
    feature.type = 'filtered', 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")`
`prompt`	boolean to turn on/off prompting user input when not plotting into a pdf-file, defaults to TRUE
`feature.type`	On which type of features should the function work? Can be either "original", "filtered", or "normalized". Please only change this paramter if you know what you are doing!
`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)

# 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. The conditional entropy evaluates associations among metadata variables and generalized linear models evaluate associations with the label, producing a correlation heatmap and appropriate quantitative barplots, respectively.

Usage

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

Arguments

`siamcat`	an object of class siamcat-class
`fn.plot`	string, filename for the pdf-plot
`meta.in`	vector, specific metadata variable names to analyze, 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`

Value

Does not return anything, but outputs plots to specified pdf file

Examples

```
# Example data
data(siamcat_example)

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

`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')
```

`create.label` *create a label object from metadata or an atomic vector*

Description

This function creates a label object from metadata or an atomic vector

Usage

```
create.label(label, case,
    meta=NULL, control=NULL,
    p.lab = NULL, n.lab = NULL,
    remove.meta.column=FALSE,
    verbose=1)
```

Arguments

`label`	named vector to create the label or the name of the metadata column that will be used to create the label
`case`	name of the group that will be used as a positive label. If the variable is binary, the other label will be used as a negative one. If the variable has multiple values, all the other values will be used a negative label (testing one vs rest).
`meta`	metadata dataframe object or an object of class sample_data-class

`control`	name of a label or vector with names that will be used as a negative label. All values that are nor equal to case and control will be dropped. Default to NULL in which case: If the variable is binary, the value not equal to case will be used as negative. If the variable has multiple values, all the values not equal to cases will be used a negative label (testing one vs rest).
`p.lab`	name of the positive group (useful mostly for visualizations). Default to NULL in which case the value of the positive group will be used.
`n.lab`	name of the negative group (useful mostly for visualizations). Default to NULL in which case the value of the negative group will be used for binary variables and "rest" will be used for variables with multiple values.
`remove.meta.column`	boolean indicating if the label column in the metadata should be retained. Please note that if this is set to `TRUE`, the function will return a list as result. Defaults to `FALSE`
`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 label-class OR a list with entries `meta` and `label`, if `remove.meta.column` is set to `TRUE`

Examples

```
data('meta_crc_zeller')
label <- create.label(label='Group', case='CRC', meta=meta.crc.zeller)
```

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

Description

Retrieve a data_split-class object from object.

Usage

```
data_split(siamcat, verbose=1)

## S4 method for signature 'ANY'
data_split(siamcat, verbose = 1)

## 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.
`verbose`	If the slot is empty, should a message be printed? values can be either 0 (no output) or 1 (print message)

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:

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

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

- `$roc.all` list of roc objects (see roc) for every repeat;
- `$auroc.all` vector of AUC values for the ROC curves for every repeat;
- `$prc.all` list of PR curves for every repeat;
- `$auprc.all` vector of AUC values for the PR curves for every repeat;
- `$ev.all` list of ev lists (see above) 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, verbose=1)

## S4 method for signature 'ANY'
eval_data(siamcat, verbose = 1)

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

Arguments

`siamcat`	(Required). An instance of siamcat-class that contains a eval_data..
`verbose`	If the slot is empty, should a message be printed? values can be either 0 (no output) or 1 (print message)

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:

- `$roc` average ROC-curve across repeats or a single ROC-curve on the complete dataset (object of class roc);
- `$auroc` AUC value for the average ROC-curve;
- `$prc` average Precision Recall curve across repeats or a single PR-curve on the complete dataset;
- `$auprc` AUC value for the average PR-curve;
- `$ev` list containing for different decision thresholds the number of false positives, false negatives, true negatives, and true positives;

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

- `$roc.all` list of roc objects (see roc) for every repeat;
- `$auroc.all` vector of AUC values for the ROC curves for every repeat;
- `$prc.all` list of PR curves for every repeat;
- `$auprc.all` vector of AUC values for the PR curves for every repeat;
- `$ev.all` list of false positive, false negatives, true negatives, true positives, and thresholds for the different repeats.

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)
```

`feat.crc.zeller` *Documentation for the example feature object in the data folder*

Description

Feature matrix (as data.frame) for the CRC dataset, containing 141 samples and 1754 bacterial species (features).

`feature_type` *Retrieve feature_type from object.*

Description

Retrieve feature_type from object.

Usage

```
feature_type(siamcat, verbose=1)

## S4 method for signature 'ANY'
feature_type(siamcat, verbose = 1)
```

Arguments

`siamcat`	(Required). An instance of siamcat-class that contains a model_list or instance of model_list-class.
`verbose`	If the slot is empty, should a message be printed? values can be either 0 (no output) or 1 (print message)

Value

The string describing type of feature used for the model or NULL.

Examples

```
    data(siamcat_example)
    feature_type(siamcat_example)
```

`feature_weights` *Retrieve feature_weights from object.*

Description

Retrieve feature_weights from object.

Usage

```
feature_weights(siamcat, verbose=1)

## S4 method for signature 'ANY'
feature_weights(siamcat, verbose = 1)
```

Arguments

`siamcat`	(Required). An instance of siamcat-class that contains a model_list or instance of model_list-class.
`verbose`	If the slot is empty, should a message be printed? values can be either 0 (no output) or 1 (print message)

Value

A dataframe containing mean/median feature weight and additional info

Examples

```
data(siamcat_example)
feature_weights(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, rm.unmapped = TRUE,
    feature.type='original', 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`
`rm.unmapped`	boolean, should unmapped reads be discarded?, defaults to `TRUE`
`feature.type`	On which type of features should the function work? Can be either "original", "filtered", or "normalized". Please only change this paramter if you know what you are doing!
`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 maximum 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.

Features can also be filtered repeatedly with different methods, e.g. first using the maximum abundance filtering and then using prevalence filtering. However, if a filtering method has already been applied to the dataset, SIAMCAT will default back on the original features for filtering.

Value

siamcat an object of class siamcat-class

Examples

```
    # Example dataset
    data(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(1:20))
```

filt_feat *Retrieve filtered features form object*

Description

Retrieve filtered features form object

Usage

```
filt_feat(siamcat, verbose=1)

## S4 method for signature 'ANY'
filt_feat(siamcat, verbose = 1)
```

Arguments

siamcat	(Required). An instance of siamcat-class that contains an object in the filt_feat slot
verbose	If the slot is empty, should a message be printed? values can be either 0 (no output) or 1 (print message)

Value

The filtered feature matrix or NULL.

Examples

```
    data(siamcat_example)
    filt_feat(siamcat_example)
```

`filt_feat-class` *The S4 class for storing the filter features/paramters*

Description

The S4 class for storing the filter features/paramters

Slots

`filt.feat` An object of class otu_table-class storing the filtered features

`filt.param` A list storing the parameters of the feature filtering

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

Description

Assign a new filt_feat object to `x`

Usage

```
filt_feat(x) <- value

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

Arguments

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

Value

none

Examples

```
data(siamcat_example)
filt_feat(siamcat_example) <- new('filt_feat',
    filt.feat=filt_feat(siamcat_example),
    filt.param=filt_params(siamcat_example))
```

filt_params *Retrieve the list of filtering parameters from object.*

Description

Retrieve the list of filtering parameters from object.

Usage

```
filt_params(siamcat, verbose=1)

## S4 method for signature 'ANY'
filt_params(siamcat, verbose = 1)
```

Arguments

siamcat	(Required). An instance of siamcat-class that contains a filt_feat object
verbose	If the slot is empty, should a message be printed? values can be either 0 (no output) or 1 (print message)

Value

The list of filtering parameters or NULL.

Examples

```
    data(siamcat_example)
    filt_params(siamcat_example)
```

get.filt_feat.matrix *get.filt_feat.matrix*

Description

Function to access features in siamcat@filt_feat@filt.feat

Usage

```
get.filt_feat.matrix(siamcat)
```

Arguments

siamcat	an object of class siamcat-class

Details

Access features in siamcat@filt_feat@filt.feat as matrix

Value

Filtered features as a matrix

Examples

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

`get.norm_feat.matrix` *get.norm_feat.matrix*

Description

Function to access features in siamcat@norm_feat@filt.feat

Usage

```
get.norm_feat.matrix(siamcat)
```

Arguments

`siamcat`	an object of class siamcat-class

Details

Access features in siamcat@norm_feat@norm.feat as matrix

Value

Normalized features as a matrix

Examples

```
data(siamcat_example)
feat <- get.norm_feat.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@phyloseq 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, verbose=1)

## S4 method for signature 'ANY'
label(siamcat, verbose = 1)

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

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

Arguments

`siamcat`	(Required). An instance of siamcat-class that contains a label or instance of label-class or a list.
`verbose`	If the slot is empty, should a message be printed? values can be either 0 (no output) or 1 (print message)

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 1 and -1
- `type` contains information about the label type
- `info` list with additional informations about the dataset

`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.crc.zeller` *Documentation for the example metadata object in the data folder*

Description

Metadata (as data.frame) for the CRC dataset, containing 6 variables (e.g. Age or BMI) for 141 samples.

`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(..., fn.plot = NULL,
    colours=NULL, verbose = 1)
```

Arguments

`...`	one or more object of class siamcat-class, can be named
`fn.plot`	string, filename for the pdf-plot
`colours`	colour specification for the different siamcat-class- objects, defaults to `NULL` which will cause the colours to be picked from the `'Set1'` palette
`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 = NULL,
    color.scheme = "BrBG",
    consens.thres = 0.5,
    heatmap.type = "zscore",
    limits = c(-3, 3), detect.lim = 1e-06,
    max.show = 50, prompt=TRUE, 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'`
`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
`prompt`	boolean to turn on/off prompting user input when not plotting into a pdf-file, 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

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, verbose=1)

## S4 method for signature 'ANY'
models(siamcat, verbose = 1)
```

Arguments

`siamcat`	(Required). An instance of siamcat-class that contains a model_list or instance of model_list-class.
`verbose`	If the slot is empty, should a message be printed? values can be either 0 (no output) or 1 (print message)

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, verbose=1)

## S4 method for signature 'ANY'
model_list(siamcat, verbose = 1)

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

## S4 method for signature 'model_list'
models(siamcat, verbose = 1)

## S4 method for signature 'model_list'
model_type(siamcat, verbose = 1)

## S4 method for signature 'model_list'
feature_type(siamcat, verbose = 1)
```

Arguments

`siamcat`	(Required). An instance of siamcat-class that contains a model_list or instance of model_list-class.
`verbose`	If the slot is empty, should a message be printed? values can be either 0 (no output) or 1 (print message)

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

`feature.type` which types of features 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, verbose=1)

## S4 method for signature 'ANY'
model_type(siamcat, verbose = 1)
```

Arguments

`siamcat`	(Required). An instance of siamcat-class that contains a model_list or instance of model_list-class.
`verbose`	If the slot is empty, should a message be printed? values can be either 0 (no output) or 1 (print message)

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),
    feature.type='filtered',
    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
`feature.type`	On which type of features should the function work? Can be either "original", "filtered", or "normalized". Please only change this paramter if you know what you are doing!
`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
- `'log.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)

# 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_feat *Retrieve normalized features form object*

Description

Retrieve normalized features form object

Usage

```
norm_feat(siamcat, verbose=1)

## S4 method for signature 'ANY'
norm_feat(siamcat, verbose = 1)
```

Arguments

siamcat	(Required). An instance of siamcat-class that contains an object in the norm_feat slot
verbose	If the slot is empty, should a message be printed? values can be either 0 (no output) or 1 (print message)

Value

The normalized feature matrix or NULL.

Examples

```
data(siamcat_example)
norm_feat(siamcat_example)
```

`norm_feat-class` *The S4 class for storing the normalization data/parameters*

Description

The S4 class for storing the normalization data/parameters

Slots

`norm.feat` An object of class otu_table-class storingthe normalized features

`norm.param` A list with:

- `norm.method` the normalization method used
- `retained.feat` the names of features retained after filtering
- `log.n0` pseudocount
- `n.p` vector norm
- `norm.margin` margin for the normalization

and additional entries depending on the normalization method used.

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

Description

Assign a new norm_feat object to `x`

Usage

```
norm_feat(x) <- value

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

Arguments

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

Value

none

Examples

```
data(siamcat_example)
norm_feat(siamcat_example) <- new("norm_feat",
    norm.feat=norm_feat(siamcat_example),
    norm.param=norm_params(siamcat_example))
```

norm_params — *Retrieve the list of normalization parameters from object.*

Description

Retrieve the list of normalization parameters from object.

Usage

```
norm_params(siamcat, verbose=1)

## S4 method for signature 'ANY'
norm_params(siamcat, verbose = 1)
```

Arguments

`siamcat`	(Required). An instance of siamcat-class that contains a norm_feat
`verbose`	If the slot is empty, should a message be printed? values can be either 0 (no output) or 1 (print message)

Value

The list of normalization parameters or NULL.

Examples

```
data(siamcat_example)
norm_params(siamcat_example)
```

orig_feat — *Retrieve a otu_table-class object from otu_table slot in the phyloseq slot in a siamcat object*

Description

Retrieve a otu_table-class object from otu_table slot in the phyloseq slot in a siamcat object

Usage

```
orig_feat(siamcat)

## S4 method for signature 'ANY'
orig_feat(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 otu_table-class.

Value

The otu_table-class object or NULL.

Examples

```
    data(siamcat_example)
    orig_feat(siamcat_example)
```

`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,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, verbose=1)

## S4 method for signature 'ANY'
physeq(siamcat, verbose = 1)

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

Arguments

`siamcat`	(Required). An instance of siamcat-class that contains a label or instance of phyloseq-class.
`verbose`	If the slot is empty, should a message be printed? values can be either 0 (no output) or 1 (print message)

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
```

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, verbose=1)

## S4 method for signature 'ANY'
pred_matrix(siamcat, verbose = 1)

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

Arguments

siamcat	(Required). An instance of siamcat-class that contains a pred_matrix
verbose	If the slot is empty, should a message be printed? values can be either 0 (no output) or 1 (print message)

Value

The pred_matrix matrix or NULL.

Examples

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

pred_matrix-class *The S4 class for storing predictions.*

Description

The S4 class for storing predictions.

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.label` *Read label 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.label(fn.in.label)
```

Arguments

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

Value

label object containing several entries:

- `$label` named vector containing the numerical labels from the file;
- `$info` information about the classes in the label;
- `$type` information about the label type (e.g. `BINARY`);

Examples

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

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

`read.lefse` *read an input file in a LEfSe input format*

Description

This reads an input file in a LEfSe input format

Usage

```
read.lefse(filename = "data.txt", rows.meta = 1, row.samples = 2)
```

Arguments

`filename`	name of the input file in a LEfSe input format
`rows.meta`	specifies in which rows medata variables are stored
`row.samples`	specifies in which row sample names are stored

Value

a list with two elements:

- `feat` a features matrix
- `meta` a metadate matrix

Examples

```
fn.in.lefse<- system.file("extdata",
"LEfSe_crc_zeller_msb_mocat_specI.tsv",package = "SIAMCAT")
meta.and.features <- read.lefse(fn.in.lefse, rows.meta = 1:6,
row.samples = 7)
meta <- meta.and.features$meta
feat <- meta.and.features$feat
label <- create.label(meta=meta, label="label", case = "cancer")
siamcat <- siamcat(feat=feat, label=label, meta=meta)
```

`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('F'), verbose=2)
## End(Not run)
```

`siamcat` *siamcat*

Description

Function to construct an object of class siamcat-class

Usage

```
siamcat(..., feat=NULL, label=NULL, meta=NULL,
    phyloseq=NULL, validate=TRUE, verbose=3)
```

Arguments

`...`	additional arguments
`feat`	feature information for SIAMCAT (see details)
`label`	label information for SIAMCAT (see details)
`meta`	(optional) metadata information for SIAMCAT (see details)
`phyloseq`	(optional) a phyloseq object for the creation of an SIAMCAT object (see details)
`validate`	boolean, should the newly constructed SIAMCAT object be validated? defaults to TRUE (**we strongly recommend against setting this parameter to FALSE**)
`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

Build siamcat-class objects from their components.

This functions creates a SIAMCAT object (see siamcat-class). In order to do so, the function needs

- feat the feature information for SIAMCAT, should be either a matrix, a data.frame, or a otu_table-class. The columns should correspond to the different samples (e.g. patients) and the rows the different features (e.g. taxa). Columns and rows should be named.
- meta metadata information for the different samples in the feature matrix. Metadata is optional for the SIAMCAT workflow. Should be either a data.frame (with the rownames corresponding to the sample names of the feature matrix) or an object of class sample_data-class
- phyloseq Alternatively to supplying both feat and meta, SIAMCAT can also work with a phyloseq object containing an otu_table and other optional slots (like sample_data for meta-variables).

Notice: do supply **either** the feature information as matrix/data.frame/otu_table (and optionally metadata) **or** a phyloseq object, but not both.

The label information for SIAMCAT can take several forms:

- metadata column: if there is metadata (either via meta or as sample_data in the phyloseq object), the label object can be created by taking the information in a specific metadata column. In order to do so, `label` should be the name of the column, and `case` should indicate which group(s) should be the positive group(s). A typical example could look like that:

  ```
  siamcat <- siamcat(feat=feat.matrix, meta=metadata,   label='DiseaseState', case='CRC')
  ```

for the construction of a label to predict CRC status (which is encoded in the column `"DiseaseState"` of the metadata). For more control (e.g. specific labels for plotting or specific control state), the label can also be created outside of the `siamcat` function using the create.label function (see below).

- named vector: the label can also be supplied as named vector which encodes the label either as characters (e.g. "Healthy" and "Diseased"), as factor, or numerically (e.g. -1 and 1). The vector must be named with the names of samples (corresponding to the samples in features). Also here, the information about the positive group(s) is needed via the `case` parameter. Internally, the vector is given to the create.label function (see for more details).
- label object: A label object can be created with the create.label function or by reading a dedicated label file with read.label.

Value

A new siamcat-class object

Examples

```
# example with package data
data("feat_crc_zeller", package="SIAMCAT")
data("meta_crc_zeller", package="SIAMCAT")

siamcat <- siamcat(feat=feat.crc.zeller,
    meta=meta.crc.zeller,
    label='Group',
    case='CRC')
```

`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

`filt_feat` an object of class filt_feat-class

`associations` an object of class associations-class

`norm_feat` an object of class norm_feat-class

`data_split` an object of class data_split-class

`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.to.lefse — *create a lefse input file from siamcat object*

Description

This function creates a lefse input file from siamcat object

Usage

```
siamcat.to.lefse(siamcat, filename = "siamcat_output.txt")
```

Arguments

`siamcat`	object of class siamcat-class
`filename`	name of the input file to which data will be save

Value

nothing but data is written to a file

Examples

```
data(siamcat_example)
siamcat.to.lefse(siamcat_example)
```

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,
    perform.fs = FALSE,
    param.fs = list(thres.fs = 100, method.fs = "AUC"),
    feature.type='normalized',
    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`
`perform.fs`	boolean, should feature selection be performed? Defaults to `FALSE`
`param.fs`	a list of parameters for the feature selection, must contain: • `thres.fs` - threshold for the feature selection, • `method.fs` - method for the feature selection, may be `AUC`, `FC`, or `Wilcoxon` See Details for more information. Defaults to `list(thres.fs=100, method.fs="AUC")`
`feature.type`	On which type of features should the function work? Can be either "original", "filtered", or "normalized". Please only change this paramter if you know what you are doing!
`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.

The function can also perform feature selection on each individual fold. At the moment, three methods for feature selection are implemented:

- `'AUC'` computes the Area Under the Receiver Operating Characteristics Curve for each single feature and selects the top `param.fs$thres.fs`, e.g. 100 features
- `'FC'` computes the generalized Fold Change (see check.associations) for each feature and likewise selects the top `param.fs$thres.fs`, e.g. 100 features
- `Wilcoxon` computes the p-Value for each single feature with the Wilcoxon test and selects features with a p-Value smaller than `param.fs$thres.fs`

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 meta-data, 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. This function is run when a siamcat-class object is created.

Value

an object of class siamcat-class with validated data

Examples

```
data(siamcat_example)
# validate.data should be run before completing the pipeline
# since the complete pipeline had been run on siamcat_example, we
# construct a new siamcat object for the example
feat <- orig_feat(siamcat_example)
label <- label(siamcat_example)
siamcat <- siamcat(feat=feat, label=label)
siamcat <- validate.data(siamcat)
```

`weight_matrix` *Retrieve weight_matrix from object.*

Description

Retrieve weight_matrix from object.

Usage

```
weight_matrix(siamcat, verbose=1)

## S4 method for signature 'ANY'
weight_matrix(siamcat, verbose = 1)
```

Arguments

`siamcat`	(Required). An instance of siamcat-class that contains a model_list or instance of model_list-class.
`verbose`	If the slot is empty, should a message be printed? values can be either 0 (no output) or 1 (print message)

Value

A matrix containing the feature weights or NULL

Examples

```
data(siamcat_example)
weight_matrix(siamcat_example)
```

Index