Package ‘scClassify’

May 16, 2024

Type Package
Title scClassify: single-cell Hierarchical Classification
Version 1.16.0
Author Yingxin Lin
Maintainer Yingxin Lin <yingxin.lin@sydney.edu.au>
Description scClassify is a multiscale classification framework for single-cell RNA-seq data based on ensemble learning and cell type hierarchies, enabling sample size estimation required for accurate cell type classification and joint classification of cells using multiple references.
License GPL-3
Encoding UTF-8
LazyData false
Depends R (>= 4.0)
Imports S4Vectors, limma, ggraph, igraph, methods, cluster, minpack.lm, mixtools, BiocParallel, proxy, proxyC, Matrix, ggplot2, hopach, diptest, mgcv, stats, graphics, statmod, Cepo
RoxygenNote 7.1.1
Suggests knitr, rmarkdown, BiocStyle, pkgdown
VignetteBuilder knitr
biocViews SingleCell, GeneExpression, Classification
BugReports https://github.com/SydneyBioX/scClassify/issues
git_url https://git.bioconductor.org/packages/scClassify
git_branch RELEASE_3_19
git_last_commit 4a3ad60
git_last_commit_date 2024-04-30
Repository Bioconductor 3.19
Date/Publication 2024-05-15
The `scClassifyTrainModel` class

**Description**

The `scClassifyTrainModel` class is designed to store training model for `scClassify`

**Usage**

```r
.scClassifyTrainModel(
  name,
  cellTypeTrain,
  cellTypeTree,
  features,
  model,
  modelweights,
  metaData
)
```
**Arguments**

- **name**: Name of the training dataset
- **cellTypeTree**: A list indicating a cell type tree
- **cellTypeTrain**: A vector of cell type in training dataset
- **features**: A vector of character indicating the features that are trained for this data
- **model**: A list storing the training model, including the features that are selected and the cell expression matrix that are used for training
- **modelweights**: A vector of numeric indicating the weights of each model
- **metaData**: A DataFrame stored meta data of training model

**Value**

A scClassifyTrainModel object

**Author(s)**

Yingxin Lin

---

### Description

Methods to access various components of the `scClassifyTrainModel` object.

### Usage

```r
cellTypeTrain(x)
```

**Arguments**

- **x**: A `scClassifyTrainModel` object.

**Value**

`cellTypeTrain` of the `scClassifyTrainModel` slot

**Examples**

```r
data(trainClassExample_xin)
cellTypeTrain(trainClassExample_xin)
```
**cellTypeTree**

*Accessors of cellTypeTree for scClassifyTrainModel*

**Description**

Methods to access various components of the `scClassifyTrainModel` object.

**Usage**

`cellTypeTree(x)`

**Arguments**

- `x` A `scClassifyTrainModel` object.

**Value**

`cellTypeTree` of the scClassifyTrainModel slot

**Examples**

```r
data(trainClassExample_xin)
cellTypeTree(trainClassExample_xin)
```

---

**features**

*Accessors of features for scClassifyTrainModel*

**Description**

Methods to access various components of the `scClassifyTrainModel` object.

**Usage**

`features(x)`

**Arguments**

- `x` A `scClassifyTrainModel` object.

**Value**

features of the scClassifyTrainModel slot

**Examples**

```r
data(trainClassExample_xin)
features(trainClassExample_xin)
```
getN

Function to get the required N given by the accuracy and the learning curve model

Description

Function to get the required N given by the accuracy and the learning curve model

Usage

getN(res, acc = 0.9)

Arguments

res         model results returned by learning_curve function
acc         accuracy that are quired

Value

sample size that are required

Examples

```r
set.seed(2019)
n <- seq(20, 10000, 100)
accMat <- do.call(cbind, lapply(1:length(n), function(i){
tmp_n <- rep(n[i], 50)
y <- -2/(tmp_n^0.8) + 0.95 + rnorm(length(tmp_n), 0, 0.02)
}))
res <- learningCurve(accMat = accMat, n)
N <- getN(res, acc = 0.9)
```

learningCurve

Fit learning curve for accuracy matrix

Description

Fit learning curve for accuracy matrix
Usage

learningCurve(
    accMat,
    n,
    auto_initial = TRUE,
    a = NULL,
    b = NULL,
    c = NULL,
    d_list = NULL,
    fitmodel = c("nls", "nls_mix", "gam"),
    plot = TRUE,
    verbose = TRUE
)

Arguments

accMat          Matrix of accuracy rate where column indicate different sample size
n               Vector indicates the sample size
auto_initial    whether automatical initialise
a               input the parameter a starting point
b               input the parameter a starting point
c               input the parameter a starting point
d_list          range of d
fitmodel        "nls", "nls_mix", "gam"
plot            indicates whether plot or not
verbose         indicates whether verbose or not

Value

list of results

Author(s)

Yingxin Lin

Examples

set.seed(2019)
n <- seq(20, 10000, 100)
accMat <- do.call(cbind, lapply(1:length(n), function(i){
tmp_n <- rep(n[i], 50)
y <- -2/(tmp_n^0.8) + 0.95 + rnorm(length(tmp_n), 0, 0.02)
}))
res <- learningCurve(accMat = accMat, n)
N <- getN(res, acc = 0.9)
**model**

*Accessors of model for scClassifyTrainModel*

**Description**
Methods to access various components of the ‘scClassifyTrainModel’ object.

**Usage**

```r
model(x)
```

**Arguments**

`x`  
A ‘scClassifyTrainModel’ object.

**Value**

model of the scClassifyTrainModel slot

**Examples**

```r
data(trainClassExample_xin)
model(trainClassExample_xin)
```

---

**modelweights**

*Accessors of modelweights for scClassifyTrainModel*

**Description**
Methods to access various components of the ‘scClassifyTrainModel’ object.

**Usage**

```r
modelweights(x)
```

**Arguments**

`x`  
A ‘scClassifyTrainModel’ object.

**Value**

modelweights of the scClassifyTrainModel slot

**Examples**

```r
data(trainClassExample_xin)
modelweights(trainClassExample_xin)
```
### name

**Accessors of name for scClassifyTrainModel**

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Methods to access various components of the <code>scClassifyTrainModel</code> object.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>name(x)</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>x</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>name of the scClassifyTrainModel slot</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>data(trainClassExample_xin)</code></td>
</tr>
<tr>
<td><code>name(trainClassExample_xin)</code></td>
</tr>
</tbody>
</table>

---

### plotCellTypeTree

**To plot cell type tree**

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>To plot cell type tree</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>plotCellTypeTree(cutree_list, group_level = NULL)</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>cutree_list</code></td>
</tr>
<tr>
<td><code>group_level</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A ggplot object visualising the HOPACH tree</td>
</tr>
</tbody>
</table>
**Examples**

```r
data("trainClassExample_xin")

plotCellTypeTree(cellTypeTree(trainClassExample_xin))
```

**Description**

Testing scClassify model

**Usage**

```r
predict_scClassify(
    exprsMat_test,
    trainRes,
    cellTypes_test = NULL,
    k = 10,
    prob_threshold = 0.7,
    cor_threshold_static = 0.5,
    cor_threshold_high = 0.7,
    features = "limma",
    algorithm = "WKNN",
    similarity = "pearson",
    cutoff_method = c("dynamic", "static"),
    weighted_ensemble = FALSE,
    weights = NULL,
    parallel = FALSE,
    BPPARAM = BiocParallel::SerialParam(),
    verbose = FALSE
)
```

**Arguments**

- `exprsMat_test` A list or a matrix indicates the log-transformed expression matrices of the query datasets
- `trainRes` A `scClassifyTrainModel` or a `list` indicates scClassify trained model
- `cellTypes_test` A list or a vector indicates cell types of the query datasets (Optional).
- `k` An integer indicates the number of neighbour
- `prob_threshold` A numeric indicates the probability threshold for KNN/WKNN/DWKNN.
- `cor_threshold_static` A numeric indicates the static correlation threshold.
- `cor_threshold_high` A numeric indicates the highest correlation threshold
features    A vector indicates the gene selection method, set as "limma" by default. This should be one or more of "limma", "DV", "DD", "chisq", "BI".

algorithm    A vector indicates the KNN method that are used, set as "WKNN" by default. This should be one or more of "WKNN", "KNN", "DWKNN".

similarity    A vector indicates the similarity measure that are used, set as "pearson" by default. This should be one or more of "pearson", "spearman", "cosine", "jaccard", "kendall", "binomial", "weighted_rank", "manhattan"

cutoff_method    A vector indicates the method to cutoff the correlation distribution. Set as "dynamic" by default.

weighted_ensemble    A logical input indicates in ensemble learning, whether the results is combined by a weighted score for each base classifier.

weights    A vector indicates the weights for ensemble

parallel    A logical input indicates whether running in parallel or not

BPPARAM    A BiocParallelParam class object from the BiocParallel package is used. Default is SerialParam().

verbose    A logical input indicates whether the intermediate steps will be printed

Value

list of results

Author(s)

Yingxin Lin

Examples

data("scClassify_example")
wang_cellTypes <- scClassify_example$wang_cellTypes
eprsMat_wang_subset <- scClassify_example$exprsMat_wang_subset
data("trainClassExample_xin")

pred_res <- predict_scClassify(exprsMat_test = exprsMat_wang_subset, 
trainRes = trainClassExample_xin, 
cellTypes_test = wang_cellTypes, 
algorithm = "WKNN", 
features = c("limma"), 
similarity = c("pearson"), 
prob_threshold = 0.7, 
verbose = TRUE)
**predict_scClassifyJoint**

*Testing scClassify model (joint training)*

**Description**

Testing scClassify model (joint training)

**Usage**

```r
predict_scClassifyJoint(
  exprsMat_test,
  trainRes,
  cellTypes_test = NULL,
  k = 10,
  prob_threshold = 0.7,
  cor_threshold_static = 0.5,
  cor_threshold_high = 0.7,
  features = "limma",
  algorithm = "WKNN",
  similarity = "pearson",
  cutoff_method = c("dynamic", "static"),
  parallel = FALSE,
  BPPARAM = BiocParallel::SerialParam(),
  verbose = FALSE
)
```

**Arguments**

- `exprsMat_test`: A list or a matrix indicates the expression matrices of the testing datasets
- `trainRes`: A `scClassifyTrainModel` or a `list` indicates scClassify training model
- `cellTypes_test`: A list or a vector indicates cell types of the testing datasets (Optional).
- `k`: An integer indicates the number of neighbour
- `prob_threshold`: A numeric indicates the probability threshold for KNN/WKNN/DWKNN.
- `cor_threshold_static`: A numeric indicates the static correlation threshold.
- `cor_threshold_high`: A numeric indicates the highest correlation threshold
- `features`: A vector indicates the method to select features, set as "limma" by default. This should be one or more of "limma", "DV", "DD", "chisq", "BI".
- `algorithm`: A vector indicates the KNN method that are used, set as "WKNN" by default. This should be one or more of "WKNN", "KNN", "DWKNN".
- `similarity`: A vector indicates the similarity measure that are used, set as "pearson" by default. This should be one or more of "pearson", "spearman", "cosine", "jaccard", "kendall", "binomial", "weighted_rank", "manhattan"
runHOPACH

Description

A function generating HOPACH tree using the average expression matrix for each cell type.

Usage

runHOPACH(data, plot = TRUE, kmax = 5)
**Arguments**

- **data**: A matrix of average expression matrix (each row indicates the gene, each column indicates the cell type)
- **plot**: Indicate whether plot or not
- **kmax**: Integer between 1 and 9 specifying the maximum number of children at each node in the tree.

**Value**

Return a list where

- **cutree_list**: A list indicates the hierarchical cell type tree
- **plot**: A `ggplot` visualise the cell type tree

**Author(s)**

Yingxin Lin

**References**


**Examples**

```r
data("scClassify_example")
wang_cellTypes <- factor(scClassify_example$wang_cellTypes)
exprsMat_wang_subset <- scClassify_example$exprsMat_wang_subset
avgMat_wang <- apply(exprsMat_wang_subset, 1, function(x)
  aggregate(x, list(wang_cellTypes), mean)$x)
rownames(avgMat_wang) <- levels(wang_cellTypes)
res_hopach <- runHOPACH(avgMat_wang)
res_hopach$plot
```

---

**runSampleCal**

Run sample size calculation for pilot data for reference dataset

**Description**

Run sample size calculation for pilot data for reference dataset
Usage

```r
runSampleCal(
  exprsMat,
  cellTypes,
  n_list = c(20, 40, 60, 80, 100, seq(200, 500, 100)),
  num_repeat = 20,
  level = NULL,
  cellType_tree = NULL,
  BPPARAM = BiocParallel::SerialParam(),
  subset_test = FALSE,
  num_test = NULL,
  ...
)
```

Arguments

- **exprsMat**: A matrix of expression matrix of pilot dataset (log-transformed, or normalised)
- **cellTypes**: A vector of cell types of pilot dataset
- **n_list**: A vector of integer indicates the sample size to run.
- **num_repeat**: An integer indicates the number of run for each sample size will be repeated.
- **level**: An integer indicates the accuracy rate is calculate based on the n-th level from top of cell type tree. If it is NULL (by default), it will be the bottom of the cell type tree. It can not be larger than the total number of levels of the tree.
- **cellType_tree**: A list indicates the cell type tree (optional), if it is NULL, the accuracy rate is calculate based on the provided cellTypes.
- **BPPARAM**: A BiocParallelParam class object from the BiocParallel package is used. Default is SerialParam().
- **subset_test**: A logical input indicates whether we used a subset of data (fixed number for each sample size) to test instead of all remaining data. By default, it is FALSE.
- **num_test**: An integer indicates the size of the test data.
- **...**: other parameter from scClassify

Value

A matrix of accuracy matrix, where columns corresponding to different sample sizes, rows corresponding to the number of repetition.

Examples

```r
data("scClassify_example")
xin_cellTypes <- scClassify_example$xin_cellTypes
exprsMat_xin_subset <- scClassify_example$exprsMat_xin_subset

exprsMat_xin_subset <- as(exprsMat_xin_subset, "dgCMatrix")
set.seed(2019)
accMat <- runSampleCal(exprsMat_xin_subset,
```
scClassify

Description
Train and test scClassify model

Usage
scClassify(
  exprsMat_train = NULL,
  cellTypes_train = NULL,
  exprsMat_test = NULL,
  cellTypes_test = NULL,
  tree = "HOPACH",
  algorithm = "WKNN",
  selectFeatures = "limma",
  similarity = "pearson",
  cutoff_method = c("dynamic", "static"),
  weighted_ensemble = FALSE,
  weights = NULL,
  weighted_jointClassification = TRUE,
  cellType_tree = NULL,
  k = 10,
  topN = 50,
  hopach_kmax = 5,
  pSig = 0.01,
  prob_threshold = 0.7,
  cor_threshold_static = 0.5,
  cor_threshold_high = 0.7,
  returnList = TRUE,
  parallel = FALSE,
  BPPARAM = BiocParallel::SerialParam(),
  verbose = FALSE
)

Arguments
exprsMat_train  A matrix of log-transformed expression matrix of reference dataset
cellTypes_train  A vector of cell types of reference dataset
eexprsMat_test  A list or a matrix indicates the expression matrices of the query datasets
cellTypes_test  A list or a vector indicates cell types of the query datasets (Optional).
tree        A vector indicates the method to build hierarchical tree, set as "HOPACH" by default. This should be one of "HOPACH" and 'HC' (using hclust).
algorithm  A vector indicates the KNN method that are used, set as "WKNN" by default. This should be one or more of "WKNN", "KNN", "DWKNN".
selectFeatures  A vector indicates the gene selection method, set as "limma" by default. This should be one or more of "limma", "DV", "DD", "chisq", "BI" and "Cepo".
similarity  A vector indicates the similarity measure that are used, set as "pearson" by default. This should be one or more of "pearson", "spearman", "cosine", "jaccard", "kendall", "binomial", "weighted_rank", "manhattan"
cutoff_method  A vector indicates the method to cutoff the correlation distribution. Set as "dynamic" by default.
weighted_ensemble  A logical input indicates in ensemble learning, whether the results is combined by a weighted score for each base classifier.
weights  A vector indicates the weights for ensemble
weighted_jointClassification  A logical input indicates in joint classification using multiple training datasets, whether the results is combined by a weighted score for each training model.
cellType_tree  A list indicates the cell type tree provided by user. (By default, it is NULL) (Only for one training data input)
k  An integer indicates the number of neighbour
topN  An integer indicates the top number of features that are selected
hopach_kmax  An integer between 1 and 9 specifying the maximum number of children at each node in the HOPACH tree.
pSig  A numeric indicates the cutoff of pvalue for features
prob_threshold  A numeric indicates the probability threshold for KNN/WKNN/DWKNN.
cor_threshold_static  A numeric indicates the static correlation threshold.
cor_threshold_high  A numeric indicates the highest correlation threshold
returnList  A logical input indicates whether the output will be class of list
parallel  A logical input indicates whether running in parallel or not
BPPARAM  A BiocParallelParam class object from the BiocParallel package is used. Default is SerialParam().
verbose  A logical input indicates whether the intermediate steps will be printed

Value

A list of the results, including testRes storing the results of the testing information, and trainRes storing the training model information.
scClassifyTrainModel-class

Author(s)

Yingxin Lin

Examples

data("scClassify_example")
xin_cellTypes <- scClassify_example$xin_cellTypes
exprsMat_xin_subset <- scClassify_example$exprsMat_xin_subset
wang_cellTypes <- scClassify_example$wang_cellTypes
exprsMat_wang_subset <- scClassify_example$exprsMat_wang_subset

scClassify_res <- scClassify(exprsMat_train = exprsMat_xin_subset,
cellTypes_train = xin_cellTypes,
exprsMat_test = list(wang = exprsMat_wang_subset),
cellTypes_test = list(wang = wang_cellTypes),
tree = "HOPACH",
algorithm = "WKNN",
selectFeatures = c("limma"),
similarity = c("pearson"),
returnList = FALSE,
verbose = FALSE)

scClassifyTrainModel-class

An S4 class to stored training model for scClassify

Description

An S4 class to stored training model for scClassify

Slots

name Name of the training dataset
cellTypeTrain A vector of cell type in training dataset
cellTypeTree A list indicate a cell type tree
features A vector of character indicates the features that are trained for this data
model A list stored the training model, including the features that are selected and the cell expression matrix that are used for training
modelweights A vector of numeric indicates the weights of each model
metaData A DataFrame stored meta data of training model
The `scClassifyTrainModelList` class

**Description**

The `scClassifyTrainModelList` class

**Usage**

`scClassifyTrainModelList(...)`

**Arguments**

... `scClassifyTrainModel` objects

**Value**

A `scClassifyTrainModelList` object

**Examples**

```r
data("trainClassExample_xin")
data("trainClassExample_wang")
trainClassExampleList <- scClassifyTrainModelList(trainClassExample_xin,
                                                  trainClassExample_wang)
```

An S4 class to stored a list of training models from `scClassify`

**Description**

An S4 class to stored a list of training models from `scClassify`
### scClassify_example

**Example data used in scClassify package**

**Description**

A list includes expression matrix and cell type of subsets of wang et al., xin et al.

**Usage**

data(scClassify_example, package = 'scClassify')

**Format**

An object of class list of length 4.

**Source**


### trainClassExample_wang

**Subset of pretrained model of Wang et al.**

**Description**

An object of scClassifyTrainModel for Wang et al.

**Usage**

data(trainClassExample_wang, package = 'scClassify')

**Format**

An object of class scClassifyTrainModel of length 1.

**Source**

trainClassExample_xin  Subset of pretrained model of Xin et al.

Description
An object of scClassifyTrainModel for Xin et al.

Usage

data(trainClassExample_xin, package = 'scClassify')

Format
An object of class scClassifyTrainModel of length 1.

Source

train_scClassify  Training scClassify model

Description
Training scClassify model

Usage

train_scClassify(
  exprsMat_train,
  cellTypes_train,
  tree = "HOPACH",
  selectFeatures = "limma",
  topN = 50,
  hopach_kmax = 5,
  pSig = 0.05,
  cellType_tree = NULL,
  weightsCal = FALSE,
  parallel = FALSE,
  BPPARAM = BiocParallel::SerialParam(),
  verbose = TRUE,
  returnList = TRUE,
  ...  
)
Arguments

exprsMat_train A matrix of log-transformed expression matrix of reference dataset
cellTypes_train A vector of cell types of reference dataset
tree A vector indicates the method to build hierarchical tree, set as "HOPACH" by default. This should be one of "HOPACH" and "HC" (using stats::hclust).
selectFeatures A vector indicates the gene selection method, set as "limma" by default. This should be one or more of "limma", "DV", "DD", "chisq", "BI", "Cepo".
topN An integer indicates the top number of features that are selected
hopach_kmax An integer between 1 and 9 specifying the maximum number of children at each node in the HOPACH tree.
pSig A numeric indicates the cutoff of pvalue for features
cellType_tree A list indicates the cell type tree provided by user. (By default, it is NULL)
weightsCal A logical input indicates whether we need to calculate the weights for the model.
parallel A logical input indicates whether the algorithms will run in parallel
BPPARAM A BiocParallelParam class object from the BiocParallel package is used. Default is SerialParam().
verbose A logical input indicates whether the intermediate steps will be printed
returnList A logical input indicates whether the output will be class of list
...
Other input for predict_scClassify for the case when weights calculation of the pretrained model is performed

Value

list of results or an object of scClassifyTrainModel

Author(s)

Yingxin Lin

Examples

data("scClassify_example")
xin_cellTypes <- scClassify_example$xin_cellTypes
exprsMat_xin_subset <- scClassify_example$exprsMat_xin_subset
trainClass <- train_scClassify(exprsMat_train = exprsMat_xin_subset,
cellTypes_train = xin_cellTypes,
selectFeatures = c("limma", "BI"),
returnList = FALSE)
Index

* datasets
  scClassify_example, 19
  trainClassExample_wang, 19
  trainClassExample_xin, 20
  .scClassifyTrainModel, 2

  cellTypeTrain, 3
  cellTypeTrain, scClassifyTrainModel-method (cellTypeTrain), 3
  cellTypeTree, 4
  cellTypeTree, scClassifyTrainModel-method (cellTypeTree), 4

  features, 4
  features, scClassifyTrainModel-method (features), 4

  getN, 5

  learningCurve, 5

  model, 7
  model, scClassifyTrainModel-method (model), 7
  modelweights, 7
  modelweights, scClassifyTrainModel-method (modelweights), 7

  name, 8
  name, scClassifyTrainModel-method (name), 8

  plotCellTypeTree, 8
  predict_scClassify, 9
  predict_scClassifyJoint, 11

  runHOPACH, 12
  runSampleCal, 13

  scClassify, 15
  scClassify_example, 19

  scClassifyTrainModel-class, 17
  scClassifyTrainModelList, 18
  scClassifyTrainModelList-class, 18

  train_scClassify, 20
  trainClassExample_wang, 19
  trainClassExample_xin, 20