Package ‘scClassify’

November 14, 2025

Type Package

Title scClassify: single-cell Hierarchical Classification

Version 1.23.0

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 devel

git_last_commit 7865ea1

git_last_commit_date 2025-10-29

Repository Bioconductor 3.23

Date/Publication 2025-11-13

Contents

.scClassifyTrainModel 2
cellTypeTrain 3
cellTypeTree 4
features 4
getN 5
learningCurve 5
model 7
modelweights 7
name 8
plotCellTypeTree 8
predict_scClassify 9
predict_scClassifyJoint 11
runHOPACH 12
runSampleCal 13
scClassify 15
scClassifyTrainModel-class 17
scClassifyTrainModelList 18
scClassifyTrainModelList-class 18
scClassify_example 19
trainClassExample_wang 19
trainClassExample_xin 20
train_scClassify 20

Index **22**

`.scClassifyTrainModel` *The scClassifyTrainModel class*

Description

The scClassifyTrainModel class is designed to stored training model for scClassify

Usage

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

Arguments

`name`	Name of the training dataset
`cellTypeTree`	A list indicate a cell type tree
`cellTypeTrain`	A vector of cell type in training dataset
`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

Value

A scClassifyTrainModel object

Author(s)

Yingxin Lin

`cellTypeTrain` *Accessors of cellTypeTrain for scClassifyTrainModel*

Description

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

Usage

```
cellTypeTrain(x)
```

Arguments

`x`	A ‘scClassifyTrainModel‘ object.

Value

cellTypeTrain of the scClassifyTrainModel slot

Examples

```
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

```
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

```
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

```
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 intialise
`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

```
model(x)
```

Arguments

`x`	A ‘scClassifyTrainModel‘ object.

Value

model of the scClassifyTrainModel slot

Examples

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

`modelweights` *Accessors of modelweights for scClassifyTrainModel*

Description

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

Usage

```
modelweights(x)
```

Arguments

`x`	A ‘scClassifyTrainModel‘ object.

Value

modelweights of the scClassifyTrainModel slot

Examples

```
data(trainClassExample_xin)
modelweights(trainClassExample_xin)
```

`name` *Accessors of name for scClassifyTrainModel*

Description

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

Usage

```
name(x)
```

Arguments

`x`	A ‘scClassifyTrainModel‘ object.

Value

name of the scClassifyTrainModel slot

Examples

```
data(trainClassExample_xin)
name(trainClassExample_xin)
```

`plotCellTypeTree` *To plot cell type tree*

Description

To plot cell type tree

Usage

```
plotCellTypeTree(cutree_list, group_level = NULL)
```

Arguments

`cutree_list`	A list indicates the hierarchical cell type tree
`group_level`	Indicate whether plot or not

Value

A ggplot object visualising the HOPACH tree

Examples

```
data("trainClassExample_xin")

plotCellTypeTree(cellTypeTree(trainClassExample_xin))
```

`predict_scClassify` *Testing scClassify model*

Description

Testing scClassify model

Usage

```
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 qurey 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 paralllel 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$wang_cellTypes
exprsMat_wang_subset <- scClassify_example$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

```
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"

`cutoff_method`	A vector indicates the method to cutoff the correlation distribution. Set as "dynamic" by default.
`parallel`	A logical input indicates whether running in paralllel 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$wang_cellTypes
exprsMat_wang_subset <- scClassify_example$exprsMat_wang_subset
data("trainClassExample_xin")
data("trainClassExample_wang")

trainClassExampleJoint <- scClassifyTrainModelList(trainClassExample_wang,
trainClassExample_xin)

pred_res_joint <- predict_scClassifyJoint(exprsMat_test = exprsMat_wang_subset,
trainRes = trainClassExampleJoint,
cellTypes_test = wang_cellTypes,
algorithm = "WKNN",
features = c("limma"),
similarity = c("pearson"),
prob_threshold = 0.7,
verbose = FALSE)

table(pred_res_joint$jointRes$cellTypes, wang_cellTypes)
```

`runHOPACH` *Create HOPACH tree*

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

van der Laan, M. J. and Pollard, K. S. (2003) ‘A new algorithm for hybrid hierarchical clustering with visualization and the bootstrap’, Journal of Statistical Planning and Inference. doi: 10.1016/S0378-3758(02)00388-9.

Examples

```
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

```
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 ogical 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 repetation.

Examples

```
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,
```

```
xin_cellTypes,
n_list = seq(20, 100, 20),
num_repeat = 5, BPPARAM = BiocParallel::SerialParam())
```

`scClassify` *Train and test scClassify model*

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
`exprsMat_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. Thisshould 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 paralllel 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 inforamtion.

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

`scClassifyTrainModelList` *The scClassifyTrainModelList class*

Description

The scClassifyTrainModelList class

Usage

```
scClassifyTrainModelList(...)
```

Arguments

`...`	scClassifyTrainModel objects

Value

A scClassifyTrainModelList object

Examples

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

`scClassifyTrainModelList-class` *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

Wang YJ, Schug J, Won K-J, Liu C, Naji A, Avrahami D, Golson ML & Kaestner KH (2016) Single cell transcriptomics of the human endocrine pancreas. Diabetes: db160405

Xin Y, Kim J, Okamoto H, Ni M, Wei Y, Adler C, Murphy AJ, Yancopoulos GD, Lin C & Gromada J (2016) RNA sequencing of single human islet cells reveals type 2 diabetes genes. Cell Metab. 24: 608–615

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

Description

An obejct of scClassifyTrainModel for Wang et al.

Usage

```
data(trainClassExample_wang, package = 'scClassify')
```

Format

An object of class `scClassifyTrainModel` of length 1.

Source

Wang YJ, Schug J, Won K-J, Liu C, Naji A, Avrahami D, Golson ML & Kaestner KH (2016) Single cell transcriptomics of the human endocrine pancreas. Diabetes: db160405

`trainClassExample_xin` *Subset of pretrained model of Xin et al.*

Description

An obejct of scClassifyTrainModel for Xin et al.

Usage

```
data(trainClassExample_xin, package = 'scClassify')
```

Format

An object of class `scClassifyTrainModel` of length 1.

Source

Xin Y, Kim J, Okamoto H, Ni M, Wei Y, Adler C, Murphy AJ, Yancopoulos GD, Lin C & Gromada J (2016) RNA sequencing of single human islet cells reveals type 2 diabetes genes. Cell Metab. 24: 608–615

`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 algorihms 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