

# Package ‘rScudo’

December 2, 2025

**Type** Package

**Title** Signature-based Clustering for Diagnostic Purposes

**Version** 1.26.0

**Description** SCUDO (Signature-based Clustering for Diagnostic Purposes) is a rank-based method for the analysis of gene expression profiles for diagnostic and classification purposes. It is based on the identification of sample-specific gene signatures composed of the most up- and down-regulated genes for that sample. Starting from gene expression data, functions in this package identify sample-specific gene signatures and use them to build a graph of samples. In this graph samples are joined by edges if they have a similar expression profile, according to a pre-computed similarity matrix. The similarity between the expression profiles of two samples is computed using a method similar to GSEA. The graph of samples can then be used to perform community clustering or to perform supervised classification of samples in a testing set.

**License** GPL-3

**Encoding** UTF-8

**LazyData** true

**URL** <https://github.com/Matteo-Ciciani/scudo>

**BugReports** <https://github.com/Matteo-Ciciani/scudo/issues>

**Collate** 'class.R' 'accessors.R' 'packageDoc.R' 'utilities.R'  
'scudoClassifyUtilities.R' 'scudoClassify.R' 'scudoTrain.R'  
'scudoCytoscape.R' 'scudoModel.R' 'scudoNetwork.R'  
'scudoPlot.R' 'scudoTest.R' 'show.R'

**RoxygenNote** 6.1.1

**Depends** R (>= 3.6)

**Imports** methods, stats, igraph, stringr, grDevices, Biobase,  
S4Vectors, SummarizedExperiment, BiocGenerics

**Suggests** testthat, BiocStyle, knitr, rmarkdown, ALL, RCy3, caret,  
e1071, parallel, doParallel

**VignetteBuilder** knitr

**biocViews** GeneExpression, DifferentialExpression,  
BiomedicalInformatics, Classification, Clustering,  
GraphAndNetwork, Network, Proteomics, Transcriptomics,  
SystemsBiology, FeatureExtraction

**git\_url** <https://git.bioconductor.org/packages/rScudo>  
**git\_branch** RELEASE\_3\_22  
**git\_last\_commit** aed3af8  
**git\_last\_commit\_date** 2025-10-29  
**Repository** Bioconductor 3.22  
**Date/Publication** 2025-12-01  
**Author** Matteo Ciciani [aut, cre],  
Thomas Cantore [aut],  
Enrica Colasurdo [ctb],  
Mario Lauria [ctb]  
**Maintainer** Matteo Ciciani <matteo.ciciani@gmail.com>

Contents

rScudo-package . . . . .	2
scudoClassify . . . . .	3
scudoCytoscape . . . . .	5
scudoModel . . . . .	6
scudoNetwork . . . . .	7
scudoPlot . . . . .	9
ScudoResults-class . . . . .	10
scudoTest . . . . .	12
scudoTrain . . . . .	13
<b>Index</b>	<b>17</b>

---

rScudo-package	<i>Signature-based Clustering for Diagnostic Purposes</i>
----------------	---

---

Description

SCUDO (Signature-based Clustering for Diagnostic Purposes) is a rank-based method for the analysis of gene expression profiles for diagnostic and classification purposes. It is based on the identification of sample-specific gene signatures composed of the most up- and down-regulated genes for that sample. Starting from gene expression data, functions in this package identify sample-specific gene signatures and use them to build a graph of samples. In this graph samples are joined by edges if they have a similar expression profile, according to a pre-computed similarity matrix. The similarity between the expression profiles of two sample is computed using a method similar to GSEA. The graph of samples can the be used to perform community clustering or to perform supervised classification of samples in a testing set.

Author(s)

Matteo Ciciani <matteo.ciciani@gmail.com>, Thomas Cantore <cantorethomas@gmail.com>

See Also

[scudoTrain](#), [scudoNetwork](#), [scudoClassify](#)

## Examples

```
# To learn more about rScudo, start with the vignette:
browseVignettes("rScudo")
```

---

scudoClassify

*Performs classification using SCUDO*


---

## Description

Performs supervised classification of samples in a testing set using a network of samples generated by SCUDO during a training step.

## Usage

```
scudoClassify(trainExpData, testExpData, N, nTop, nBottom,
  trainGroups, maxDist = 1, weighted = TRUE, complete = FALSE, beta = 1,
  alpha = 0.1, foldChange = TRUE, featureSel = TRUE, logTransformed = NULL,
  parametric = FALSE, pAdj = "none", distFun = NULL)
```

## Arguments

trainExpData	either an <a href="#">ExpressionSet</a> , a <a href="#">SummarizedExperiment</a> , a data.frame or a matrix of gene expression data, with a column for each sample and a row for each feature
testExpData	either an <a href="#">ExpressionSet</a> , a <a href="#">SummarizedExperiment</a> , a data.frame or a matrix of gene expression data, with a column for each sample and a row for each feature
N	a number between 0 and 1, representing the fraction of the signature-to-signature distances that will be used to draw the graph
nTop	number of up-regulated features to include in the signatures
nBottom	number of down-regulated features to include in the signatures
trainGroups	factor containing group labels for each sample in trainExpData
maxDist	an integer. Only nodes with a distance from a testing node less or equal to maxDist are used to perform the classification
weighted	logical, whether to consider the distances associated to the edges to compute the scores for the classification. For a description of the classification method, see <a href="#">Details</a> below
complete	logical, whether to consider all the nodes in the training set to perform the classification. If TRUE, the arguments N, maxDist, weighted and beta are ignored. For a description of the classification method, see <a href="#">Details</a> below
beta	a coefficient used to down-weight the influence of distant nodes on the classification outcome. For a description of the classification method, see <a href="#">Details</a> below
alpha	p-value cutoff for the optional feature selection step. If feature selection is skipped, alpha is ignored
foldChange	logical, whether or not to compute fold-changes from expression data

<code>featureSel</code>	logical, whether or not to perform a feature selection. Feature selection is performed using one of four tests: Student's t-test, ANOVA, Wilcoxon-Mann-Withney test, or Kruskal-Wallis test. The test used depends on the number of groups and the parametric argument
<code>logTransformed</code>	logical or NULL. It indicates whether the data is log-transformed. If NULL, an attempt is made to guess if the data is log-transformed
<code>parametric</code>	logical, whether to use a parametric or a non-parametric test for the feature selection
<code>pAdj</code>	<code>pAdj</code> method to use to adjust the p-values in the feature selection step. See <a href="#">p.adjust.methods</a> for a list of adjustment methods
<code>distFun</code>	the function used to compute the distance between two samples. See Details of <a href="#">scudoTrain</a> for the specification of this function

### Details

This function performs supervised classification of samples in a testing set, using the networks similar to the one generated by [scudoTrain](#) and [scudoNetwork](#) as a model.

For each sample *S* in the testing set, a new distance matrix is computed using the expression profiles in the training set and the expression profile of *S*. The distance matrix is computed as described in the Details of [scudoTrain](#).

If the argument `complete` is TRUE, the distance matrix is converted in a similarity score matrix. Then, the similarity scores between *S* and all the samples in the training set are aggregated according to groups. The mean similarity scores are computed for each group and classification scores are generated dividing them by their sum, obtaining values between 0 and 1.

If the argument `complete` is FALSE, the distance matrix obtained from *S* and the training set is used to generate a network of samples, using the parameter `N` as a threshold for edge selection (see Details of [scudoNetwork](#) for a more complete description). Then the neighbors of *S* in the network are explored, up to a distance controlled by the parameter `maxDist`. If the `weighted` parameter is FALSE, the classification scores for each group are computed as the number of edges connecting *S* or one of its neighbors to a node of that group. The scores are then rescaled dividing them by their sum, in order to obtain values between 0 and 1. If the `weighted` parameter is TRUE, the classification scores for each group are computed as the sum of the similarity scores associated to edges connecting *S* or one of its neighbors to nodes of that group. The scores are then rescaled dividing them by their sum, in order to obtain values between 0 and 1. The parameter `beta` can be used to down-weight the contribution to the classification scores of edges connecting nodes distant from *S*, both in the weighed and unweighted cases.

The predicted group for each sample is the one with the largest classification score. Both predictions and classification scores are returned. Note that if the argument `complete` is FALSE, the classification scores for a sample may be all zero, which happens when the corresponding node is isolated in the network of samples. In this case the predicted group is NA. The tuning of the parameters can be performed automatically using the [train](#) function from the package `caret` and the function [scudoModel](#).

### Value

A list containing a factor with the predicted class for each sample in `testExpData` and a data.frame of the classification scores used to generate the predictions.

### Author(s)

Matteo Ciciani <matteo.ciciani@gmail.com>, Thomas Cantore <cantorethomas@gmail.com>

**See Also**

[scudoTrain](#), [scudoModel](#)

**Examples**

```
expData <- data.frame(a = 1:10, b = 2:11, c = 10:1, d = 11:2,
  e = c(1:4, 10:5), f = c(7:10, 6:1), g = c(8:4, 1:3, 10, 9),
  h = c(6:10, 5:1), i = c(5:1, 6:10))
rownames(expData) <- letters[1:10]
groups <- factor(c(1,1,1,2,2,2,2,1,1,1))
inTrain <- 1:5

# perform classification
res <- scudoClassify(expData[, inTrain], expData[, -inTrain], 0.9, 3, 3,
  groups[inTrain], featureSel = FALSE)

#explore predictions
predictions <- res$predicted
scores <- res$scores
```

---

scudoCytoscape

---

*Create a Cytoscape network from the output of scudoNetwork*


---

**Description**

A wrapper to [RCy3](#) function calls to plot the result of [scudoNetwork](#) in Cytoscape 3. Cytoscape must be open before running this function.

**Usage**

```
scudoCytoscape(graph, title = "Scudo Graph", collection = "SCUDO")
```

**Arguments**

graph	object of class <a href="#">igraph</a> , like the result of <a href="#">scudoNetwork</a>
title	the title of the network
collection	the name of the Cytoscape collection

**Value**

The network SUID (an integer).

**Author(s)**

Matteo Ciciani <matteo.ciciani@gmail.com>, Thomas Cantore <cantorethomas@gmail.com>

**See Also**

[scudoNetwork](#), [RCy3](#)

## Examples

```
# generate dummy dataset
exprData <- data.frame(a = 11:20, b = 16:25,
  c = rev(1:10), d = c(1:2, rev(3:10)))
rownames(exprData) <- letters[11:20]
grps <- as.factor(c("G1", "G1", "G2", "G2"))
nTop <- 2
nBottom <- 3

# run scudoTrain and scudoNetwork
res <- scudoTrain(exprData, grps, nTop, nBottom, foldChange = FALSE,
  featureSel = FALSE)
col <- c("#FF00FF", "#FF00FF", "#00FF00", "#00FF00")
net <- scudoNetwork(res, N = 0.5, colors = col)

# run scudoCytoscape (with Cytoscape open)
## Not run: scudoCytoscape(res, title = "scudoCytoscape output")
```

---

scudoModel

*Generate model for caret::train*


---

## Description

This function generates a suitable input for the method argument of the function `train` from the package `caret`, that can be used to perform automatic parameter tuning (e.g. using cross-validation).

## Usage

```
scudoModel(nTop, nBottom, N, maxDist = 1, weighted = TRUE,
  complete = FALSE, beta = 1, distFun = NULL)
```

## Arguments

nTop	number of up-regulated features to include in the signatures
nBottom	number of down-regulated features to include in the signatures
N	a number between 0 and 1, representing the fraction of the signature-to-signature distances that will be used to draw the graph
maxDist	an integer. Only nodes with a distance from a testing node less or equal to maxDist are used to perform the classification
weighted	logical, whether to consider the distances associated to the edges to compute the scores for the classification
complete	logical, whether to consider all the nodes in the training set to perform the classification. If TRUE, the arguments N, maxDist, weighted and beta are ignored
beta	a coefficient used to down-weight the influence of distant nodes on the classification outcome
distFun	the function used to compute the distance between two samples. See Details for the specification of this function

## Details

This function can be used in conjunction with the functions in the package `caret` to tune the parameters of `scudoClassify`. The input of this function are vector of parameter values that the tuning procedure should explore. All possible combination of parameter values are explored by default. The user can change this using the search argument of the `trainControl` function.

The output of this function is a list that represents a classification model using `scudoClassify` and that can be used as input for the method argument of the function `train`.

## Value

A named list

## Author(s)

Matteo Ciciani <matteo.ciciani@gmail.com>, Thomas Cantore <cantorethomas@gmail.com>

## See Also

`scudoClassify`, `train`, `trainControl`

## Examples

```
# Generate example dataset
expData <- data.frame(a = 1:10, b = 2:11, c = 10:1, d = 11:2,
  e = c(1:4, 10:5), f = c(7:10, 6:1), g = c(8:4, 1:3, 10, 9),
  h = c(6:10, 5:1), i = c(5:1, 6:10))
rownames(expData) <- letters[1:10]
groups <- factor(c(1,1,1,2,2,2,1,1,1))

# Run bootstrap. Notice that the dataset is transposed
ctrl <- caret::trainControl(method = "boot", number = 5)
model <- scudoModel(nTop = 3:5, nBottom = 3:5, N = 0.5, complete = TRUE)
set.seed(1)
bootRes <- caret::train(x = t(expData), y = groups, method = model,
  trControl = ctrl)
```

---

scudoNetwork

*Create graph from a ScudoResults object*

---

## Description

A function to create an `igraph` object from a `ScudoResults` object. In the graph, nodes are samples and edges quantify the similarity between the nodes.

## Usage

```
scudoNetwork(object, N, colors = character())
```

## Arguments

object	a <a href="#">ScudoResults</a> object
N	a number between 0 and 1, representing the fraction of the signature-to-signature distances that will be used to draw the graph
colors	a character vector of hexadecimal RGB color codes used to color the nodes of the graph. <code>length(colors)</code> must be equal to the number of samples. By default colors are chosen according to the groups in object

## Details

This function uses the distance matrix in the [ScudoResults](#) object to generate an [igraph](#) object, representing a graph where nodes are samples and edges quantify the similarity between the signatures of pairs of nodes.

The distance matrix in object is used to generate an unweighted adjacency matrix, that is then used to generate the graph. The sample quantile of probability N, computed from all the non-zero distances in the distance matrix, is used as a threshold to generate the adjacency matrix: all the distances larger than this quantile are mapped to 0, all the distances smaller than this quantile are mapped to 1 (with the exception of the distances of any node from itself, which are equal to 0).

Distances are set as attributes of the edges. Use `igraph::E(igraphObject)$distance` to retrieve them, where `igraphObject` is the result of `scudoNetwork`.

The color parameter controls the color of the nodes. It must be a vector of hexadecimal RGB color codes (like `"#FFFFFF"`), with length equal to the number of samples in object. By default, a different color is assigned to each group. If no group is specified in object, all nodes are set to the same color. A vector of node colors can be accessed with `igraph::V(igraphObject)$color`. Use `igraph::V(igraphObject)$group` to access the group label of each node (it returns NULL if no group is specified in object).

## Value

An object of class [igraph](#).

## Author(s)

Matteo Ciciani <matteo.ciciani@gmail.com>, Thomas Cantore <cantorethomas@gmail.com>

## See Also

[scudoCytoscape](#), [ScudoResults](#), [igraph](#)

## Examples

```
# generate dummy dataset and run scudo
exprData <- data.frame(a = 11:20, b = 16:25,
  c = rev(1:10), d = c(1:2, rev(3:10)))
rownames(exprData) <- letters[11:20]
grps <- as.factor(c("G1", "G1", "G2", "G2"))
nTop <- 2
nBottom <- 3

res <- scudoTrain(exprData, grps, nTop, nBottom, foldChange = FALSE,
  featureSel = FALSE)
```



```
# generate network
col <- c("#FF0000", "#FF0000", "#0000FF", "#0000FF")
net <- scudoNetwork(res, N = 0.5, colors = col)

# retrieve node colors and groups
nodes <- igraph::V(net)
colors <- nodes$color
groups <- nodes$group

# retrieve distances from edges
edges <- igraph::E(net)
dist <- edges$distance

# plot the network
scudoPlot(net)
```

---

scudoPlot

*Plot scudoNetwork result*


---

## Description

A wrapper to [plot.igraph](#) and [legend](#). Can be used to plot the result of [scudoNetwork](#) with a color legend.

## Usage

```
scudoPlot(net, x = "bottomright", y = NULL, ...)
```

## Arguments

net	an <a href="#">igraph</a> object returned by <a href="#">scudoNetwork</a>
x, y	the x and y coordinates to be used to position the legend. They can be specified by keyword or in any way which is accepted by <a href="#">xy.coords</a> . See Details of <a href="#">legend</a>
...	arguments to be passed to <a href="#">plot.igraph</a>

## Value

Returns NULL, invisibly.

## Author(s)

Matteo Ciciani <matteo.ciciani@gmail.com>, Thomas Cantore <cantorethomas@gmail.com>

## See Also

[scudoNetwork](#), [plot.igraph](#)

## Examples

```
# generate dummy dataset, run scudoTrain and scudoNetwork
exprData <- data.frame(a = 11:20, b = 16:25,
  c = rev(1:10), d = c(1:2, rev(3:10)))
rownames(exprData) <- letters[11:20]
grps <- as.factor(c("G1", "G1", "G2", "G2"))
nTop <- 2
nBottom <- 3
res <- scudoTrain(exprData, grps, nTop, nBottom, foldChange = FALSE,
  featureSel = FALSE)
net <- scudoNetwork(res, N = 0.5)

# Plot with scudoPlot
scudoPlot(net)
```

---

ScudoResults-class	<i>Class ScudoResults</i>
--------------------	---------------------------

---

## Description

This is an S4 class that represents the output of the functions [scudoTrain](#) and [scudoTest](#).

## Details

This class provides a structure to represent the results of [scudoTrain](#) and [scudoTest](#). It contains the distance matrix and the gene signatures generated by the SCUDO analysis. It is possible, although not recommended, to manually create instances of this class (see Examples below).

## Slots

**distMatrix** a symmetric matrix with non-negative numeric elements

**upSignatures** a data.frame with the same colnames as **distMatrix**, representing the up-regulated features in each sample

**downSignatures** a data.frame with the same colnames as **distMatrix**, representing the down-regulated features in each sample

**groupsAnnotation** a factor that represents the groups used for the [computeFC](#) and the feature selection

**consensusUpSignatures** a data.frame that contains the consensus signatures of up-regulated features for each group

**consensusDownSignatures** a data.frame that contains the consensus signatures of down-regulated features for each group

**selectedFeatures** a character vector of selected features. If the feature selection was not performed, it contains every feature present in the input of the [scudo](#) functions

**scudoParams** a list of the parameters used to run the function that created the instance of the class

## Methods

`distMatrix` signature(object = "ScudoResults"): a method for obtaining the distance matrix.

`upSignatures` signature(object = "ScudoResults"): a method for obtaining the signature of up-regulated features in each sample.

`downSignatures` signature(object = "ScudoResults"): a method for obtaining the signature of down-regulated features in each sample.

`groupsAnnotation` signature(object = "ScudoResults"): a method for obtaining the groups used for computeFC and feature selection.

`consensusUpSignatures` signature(object = "ScudoResults"): a method for obtaining the consensus signatures of up-regulated features in each group.

`consensusDownSignatures` signature(object = "ScudoResults"): a method for obtaining the consensus signatures of down-regulated features in each group.

`selectedFeatures` signature(object = "ScudoResults"): a method for obtaining the names of the features selected. If no feature selection was performed, the names of every feature are returned.

`scudoParams` signature(object = "ScudoResults"): a method for obtaining the parameters that were used to generate the result.

## Author(s)

Matteo Ciciani <matteo.ciciani@gmail.com>, Thomas Cantore <cantorethomas@gmail.com>

## Examples

```
# manually generate instance of ScudoResults class
m <- matrix(1, ncol = 4, nrow = 4)
diag(m) <- 0
rownames(m) <- colnames(m) <- letters[1:4]
SigUp <- data.frame(a = letters[1:5], b = letters[6:10], c = letters[11:15],
  d = letters[16:20], stringsAsFactors = FALSE)
SigDown <- data.frame(a = letters[1:10], b = letters[11:20],
  c = letters[1:10], d = letters[11:20],
  stringsAsFactors = FALSE)
groups <- as.factor(c("G1", "G1", "G2", "G2"))
ConsUp <- data.frame(G1 = letters[11:15], G2 = letters[21:25],
  stringsAsFactors = FALSE)
ConsDown <- data.frame(G1 = letters[16:25], G2 = letters[1:10],
  stringsAsFactors = FALSE)
Feats <- letters[1:20]
Pars <- list()

scudoR <- ScudoResults(distMatrix = m,
  upSignatures = SigUp,
  downSignatures = SigDown,
  groupsAnnotation = groups,
  consensusUpSignatures = ConsUp,
  consensusDownSignatures = ConsDown,
  selectedFeatures = Feats,
  scudoParams = Pars)
```

---

scudoTest

*Performs SCUDO analysis on test dataset*


---

## Description

A function to perform the SCUDO analysis on test data, given an object of class `ScudoResults` used as training model.

## Usage

```
scudoTest(trainScudoRes, testExpData, testGroups = NULL, nTop = NULL,
          nBottom = NULL, foldChange = TRUE, groupedFoldChange = FALSE,
          logTransformed = NULL, distFun = NULL)
```

## Arguments

<code>trainScudoRes</code>	an object of class <code>ScudoResult</code> used as training model
<code>testExpData</code>	either an <a href="#">ExpressionSet</a> , a <a href="#">SummarizedExperiment</a> , a <code>data.frame</code> or a matrix of gene expression data, with a column for each sample and a row for each feature
<code>testGroups</code>	factor containing group labels for each sample in <code>testExpData</code>
<code>nTop</code>	number of up-regulated features to include in the signatures. If <code>NULL</code> , it defaults to the value present in <code>trainScudoRes</code>
<code>nBottom</code>	number of down-regulated features to include in the signatures. If <code>NULL</code> , it defaults to the value present in <code>trainScudoRes</code>
<code>foldChange</code>	logical, whether or not to compute fold-changes from expression data
<code>groupedFoldChange</code>	logical, whether or not to take into account the groups when computing fold-changes. See Details for a description of the computation of fold-changes
<code>logTransformed</code>	logical or <code>NULL</code> . It indicates whether the data is log-transformed. If <code>NULL</code> , an attempt is made to guess if the data is log-transformed
<code>distFun</code>	the function used to compute the distance between two samples. See Details of <a href="#">scudoTrain</a> for the specification of this function

## Details

Given an object of class [ScudoResults](#) and a set of expression profiles with unknown classification, `scudoTest` performs an analysis similar to [scudoTrain](#), computing a list of signatures composed of genes over- and under-expressed in each sample, consensus signatures for each group and a distance matrix that quantifies the similarity between the signatures of pairs of samples.

`scudoTest` differs from `scudoTrain` in the feature selection step: only the features present in the `ScudoResults` object taken as input are considered for the following steps. The computation of fold-changes, the identification of gene signatures and the computation of the distance matrix are performed as described in the Details of [scudoTrain](#).

If the classification of samples in the testing dataset is provided, it is only used for annotation purposes.

**Value**

Object of class [ScudoResults](#).

**Author(s)**

Matteo Ciciani <matteo.ciciani@gmail.com>, Thomas Cantore <cantorethomas@gmail.com>

**See Also**

[scudoTrain](#), [scudoNetwork](#), [ScudoResults](#), [scudoClassify](#)

**Examples**

```
# generate dummy train dataset
exprDataTrain <- data.frame(a = 11:20, b = 16:25,
  c = rev(1:10), d = c(1:2, rev(3:10)))
exprDataTest <- data.frame(e = 1:10, f = 11:20,
  g = rev(11:20), h = c(1:2, rev(3:10)))
rownames(exprDataTrain) <- rownames(exprDataTest) <- letters[11:20]
grpsTrain <- as.factor(c("G1", "G1", "G2", "G2"))
nTop <- 2
nBottom <- 3

# run scudo
res <- scudoTrain(exprDataTrain, grpsTrain, nTop, nBottom,
  foldChange = FALSE, featureSel = FALSE)
show(res)

# run scudoTest
testRes <- scudoTest(res, exprDataTest, foldChange = FALSE)
show(testRes)
```

---

scudoTrain

*Performs SCUDO analysis*


---

**Description**

SCUDO (Signature-based Clustering for Diagnostic purposes) is a rank-based method for the analysis of gene expression profiles. This function computes gene signatures for each sample and consensus signatures for each group specified. A distance matrix is also computed, that can be used by the function [scudoNetwork](#) to generate a graph in which each node is a sample and an edge between two nodes quantitatively represents the similarity between their respective signatures.

**Usage**

```
scudoTrain(expressionData, groups, nTop, nBottom, alpha = 0.1,
  foldChange = TRUE, groupedFoldChange = FALSE, featureSel = TRUE,
  logTransformed = NULL, parametric = FALSE, pAdj = "none", distFun = NULL)
```

## Arguments

expressionData	either an <a href="#">ExpressionSet</a> , a <a href="#">SummarizedExperiment</a> , a data.frame or a matrix of gene expression data, with a column for each sample and a row for each feature
groups	factor containing group labels for each sample in expressionData
nTop	number of up-regulated features to include in the signatures
nBottom	number of down-regulated features to include in the signatures
alpha	p-value cutoff for the optional feature selection step. If feature selection is skipped, alpha is ignored
foldChange	logical, whether or not to compute fold-changes from expression data
groupedFoldChange	logical, whether or not to take into account the groups when computing fold-changes. See Details for a description of the computation of fold-changes
featureSel	logical, whether or not to perform a feature selection. Feature selection is performed using one of four tests: Student's t-test, ANOVA, Wilcoxon-Mann-Whitney test, or Kruskal-Wallis test. The test used depends on the number of groups and the parametric argument
logTransformed	logical or NULL. It indicates whether the data is log-transformed. If NULL, an attempt is made to guess if the data is log-transformed
parametric	logical, whether to use a parametric or a non-parametric test for the feature selection
pAdj	pAdj method to use to adjust the p-values in the feature selection step. See <a href="#">p.adjust.methods</a> for a list of adjustment methods
distFun	the function used to compute the distance between two samples. See Details for the specification of this function

## Details

Given a set of expression profiles with known classification, *scudoTrain* computes a list of signatures composed of genes over- and under-expressed in each sample. It also compute consensus signatures for each group and uses the signatures to compute a distance matrix that quantifies the similarity between the signatures of pairs of samples.

Before computing the signatures, two optional preprocessing steps are performed. In the first step fold-changes are computed from expression values. If the parameter `groupedFoldChange` is `TRUE`, the fold-changes are computed in two steps: first the mean expression value for each feature in each group is computed. Then, the fold-changes for each feature are computed dividing the expression values by the mean of the group means. If the parameter `groupedFoldChange` is `FALSE`, the fold-changes are computed dividing the expression value of each feature by the mean expression value of that feature (regardless of groups). If the expression values are log-transformed, subtraction is used instead of division.

The second optional preprocessing step is a feature selection. This step is performed in order to select relevant features. Feature selection is performed using one of four tests: Student's t-test, ANOVA, Wilcoxon-Mann-Whitney test, or Kruskal-Wallis test. The test used depends on the number of groups and the parameter `parametric`. The parameter `pAdj` controls the method used to adjust p-values for multiple hypothesis testing. For a list of adjustment methods see [p.adjust](#). Features with an adjusted p-value less than `alpha` are selected.

After these two optional steps, the signatures for each sample are computed. Selected features are ranked according to the expression values (or the fold-change, if computed). Then the first

`nTop` and the last `nBottom` features are selected from the ranked list of features. Two `data.frames` are containing the signatures of up-regulated genes and down-regulated genes for each sample are produced and are contained in the returned object.

Consensus top and bottom signatures are computed for each group. The average rank for each gene is computed for each group. Features are then ranked according to the average rank in each group and the first `nTop` and the last `nBottom` genes are selected to form the consensus signatures of each group. Two `data.frames` containing the consensus signatures are produced and are contained in the returned object.

Gene signatures are used to compute an all-to-all distance matrix. The distance between two samples quantifies the degree of similarity between the signatures of the two samples. The default method used to compute the distance between two samples is based on GSEA. Specifically, the distance between two samples A and B is computed in three steps. First the enrichment score (ES) of the signature of sample A against the whole expression profile of sample B,  $ES(A, B)$ , is computed.  $ES(B, A)$  is also computed. Since a signature is composed of a top and a bottom part, the ES of a signature in a profile is computed as the average of the ES of the top and the bottom signatures. Then, the distance between two samples is computed as the average ES:

$$d(A, B) = (ES(A, B) + ES(B, A))/2$$

Finally, a rounded value of the minimum non-zero distance is subtracted from all values; the purpose of this transformation is to expand the dynamic range and increase the relative difference between distance values.

The ES employed by default is also known as the Kolmogorov-Smirnov running sum and is analogous to the ES used in the unweighted early version of GSEA. Alternatively, a user specified function can be used to compute the distance matrix, provided using the parameter `distFun`. This function should be of the form `function(expressionData, nTop, nBottom)`, where `expressionData` is a `data.frame` of expression profiles and `nTop` and `nBottom` are the sizes of the signatures. This function should return a symmetric square matrix, with identical names on the rows and the columns, corresponding to the names of the samples in `expressionData`.

The distance matrix is included in the returned object and can be used to generate a graph of samples using [scudoNetwork](#).

Note that we use the term distance loosely: from a mathematical point of view, our "distance" is actually a semimetric (it does not satisfy the triangle inequality).

## Value

Object of class [ScudoResults](#).

## Author(s)

Matteo Ciciani <matteo.ciciani@gmail.com>, Thomas Cantore <cantorethomas@gmail.com>

## See Also

[scudoTest](#), [scudoNetwork](#), [ScudoResults](#)

## Examples

```
# generate dummy dataset
exprData <- data.frame(a = 11:20, b = 16:25,
  c = rev(1:10), d = c(1:2, rev(3:10)))
rownames(exprData) <- letters[11:20]
grps <- as.factor(c("G1", "G1", "G2", "G2"))
```

```
nTop <- 2
nBottom <- 3

# run scudo
res <- scudoTrain(exprData, grps, nTop, nBottom, foldChange = FALSE,
  featureSel = FALSE)
show(res)

# examine top signatures and top consensus signatures
upSignatures(res)
consensusUpSignatures(res)

# examine distance matrix
distMatrix(res)
```



# Index

SummarizedExperiment, [3](#), [12](#), [14](#)

consensusDownSignatures  
(ScudoResults-class), [10](#)

consensusDownSignatures, ScudoResults-method  
(ScudoResults-class), [10](#)

consensusUpSignatures  
(ScudoResults-class), [10](#)

consensusUpSignatures, ScudoResults-method  
(ScudoResults-class), [10](#)

distMatrix (ScudoResults-class), [10](#)

distMatrix, ScudoResults-method  
(ScudoResults-class), [10](#)

downSignatures (ScudoResults-class), [10](#)

downSignatures, ScudoResults-method  
(ScudoResults-class), [10](#)

ExpressionSet, [3](#), [12](#), [14](#)

groupsAnnotation (ScudoResults-class),  
[10](#)

groupsAnnotation, ScudoResults-method  
(ScudoResults-class), [10](#)

igraph, [5](#), [7–9](#)

legend, [9](#)

p.adjust, [14](#)

p.adjust.methods, [4](#), [14](#)

plot.igraph, [9](#)

RCy3, [5](#)

rScudo-package, [2](#)

scudoClassify, [2](#), [3](#), [7](#), [13](#)

scudoCytoscape, [5](#), [8](#)

scudoModel, [4](#), [5](#), [6](#)

scudoNetwork, [2](#), [4](#), [5](#), [7](#), [9](#), [13](#), [15](#)

scudoNetwork, ScudoResults-method  
(scudoNetwork), [7](#)

scudoParams (ScudoResults-class), [10](#)

scudoParams, ScudoResults-method  
(ScudoResults-class), [10](#)

scudoPlot, [9](#)

ScudoResults, [7](#), [8](#), [12](#), [13](#), [15](#)

ScudoResults (ScudoResults-class), [10](#)

ScudoResults-class, [10](#)

scudoTest, [10](#), [12](#), [15](#)

scudoTrain, [2](#), [4](#), [5](#), [10](#), [12](#), [13](#), [13](#)

selectedFeatures (ScudoResults-class),  
[10](#)

selectedFeatures, ScudoResults-method  
(ScudoResults-class), [10](#)

show, ScudoResults-method  
(ScudoResults-class), [10](#)

train, [4](#), [6](#), [7](#)

trainControl, [7](#)

upSignatures (ScudoResults-class), [10](#)

upSignatures, ScudoResults-method  
(ScudoResults-class), [10](#)

xy.coords, [9](#)