

# Package ‘scDblFinder’

January 18, 2021

**Type** Package

**Title** scDblFinder

**Version** 1.4.0

**Depends** R (>= 4.0)

**URL** <https://github.com/plger/scDblFinder>

**BugReports** <https://github.com/plger/scDblFinder/issues>

**Description** The scDblFinder package gathers various methods for the detection and handling of doublets/multiplets in single-cell RNA sequencing data (i.e. multiple cells captured within the same droplet or reaction volume). It includes methods formerly found in the scran package, and the new fast and comprehensive scDblFinder method.

**License** GPL-3

**Imports** igraph, Matrix, BiocGenerics, BiocParallel, BiocNeighbors, BiocSingular, S4Vectors, SummarizedExperiment, SingleCellExperiment, scran, scater, scuttle, bluster, methods, DelayedArray, xgboost, stats, utils

**Suggests** BiocStyle, knitr, rmarkdown, testthat, scRNAseq, bluster

**VignetteBuilder** knitr

**Encoding** UTF-8

**RoxygenNote** 7.1.1

**biocViews** Preprocessing, SingleCell, RNASeq

**git\_url** <https://git.bioconductor.org/packages/scDblFinder>

**git\_branch** RELEASE\_3\_12

**git\_last\_commit** b3dd92a

**git\_last\_commit\_date** 2020-10-27

**Date/Publication** 2021-01-17

**Author** Pierre-Luc Germain [cre, aut] (<<https://orcid.org/0000-0003-3418-4218>>), Aaron Lun [ctb]

**Maintainer** Pierre-Luc Germain <[pierre-luc.germain@hest.ethz.ch](mailto:pierre-luc.germain@hest.ethz.ch)>

**R topics documented:**

addDoublets . . . . .	2
computeDoubletDensity . . . . .	3
createDoublets . . . . .	6
cxds2 . . . . .	7
data . . . . .	7
doubletThresholding . . . . .	8
fastcluster . . . . .	8
findDoubletClusters . . . . .	9
getArtificialDoublets . . . . .	12
getCellPairs . . . . .	13
getExpectedDoublets . . . . .	14
mockDoubletSCE . . . . .	14
plotDoubletMap . . . . .	15
recoverDoublets . . . . .	16
scDbfFinder . . . . .	18
selFeatures . . . . .	21
<b>Index</b>	<b>22</b>

---

addDoublets	<i>addDoublets</i>
-------------	--------------------

---

**Description**

Adds artificial doublets to an existing dataset

**Usage**

```
addDoublets(
  x,
  clusters,
  dbr = (0.01 * ncol(x)/1000),
  only.heterotypic = TRUE,
  adjustSize = FALSE,
  prefix = "doublet.",
  ...
)
```

**Arguments**

x	A count matrix of singlets, or a <a href="#">SummarizedExperiment-class</a>
clusters	A vector of cluster labels for each column of 'x'
dbr	The doublet rate
only.heterotypic	Whether to add only heterotypic doublets.
adjustSize	Whether to adjust the library sizes of the doublets.
prefix	Prefix for the colnames generated.
...	Any further arguments to <a href="#">createDoublets</a> .

**Value**

A 'SingleCellExperiment' with the colData columns 'cluster' and 'type' (indicating whether the cell is a singlet or doublet).

**Examples**

```
sce <- mockDoubletSCE(dbl.rate=0)
sce <- addDoublets(sce, clusters=sce$cluster)
```

---

computeDoubletDensity *Compute the density of simulated doublets*

---

**Description**

Identify potential doublet cells based on the local density of simulated doublet expression profiles. This replaces the older doubletCells function from the **scran** package.

**Usage**

```
computeDoubletDensity(x, ...)

## S4 method for signature 'ANY'
computeDoubletDensity(
  x,
  size.factors.norm = NULL,
  size.factors.content = NULL,
  k = 50,
  subset.row = NULL,
  niters = max(10000, ncol(x)),
  block = 10000,
  dims = 25,
  BNPARAM = KmknParam(),
  BSPARAM = bsparam(),
  BPPARAM = SerialParam()
)

## S4 method for signature 'SummarizedExperiment'
computeDoubletDensity(x, ..., assay.type = "counts")

## S4 method for signature 'SingleCellExperiment'
computeDoubletDensity(x, size.factors.norm = sizeFactors(x), ...)
```

**Arguments**

x	A numeric matrix-like object of count values, where each column corresponds to a cell and each row corresponds to an endogenous gene. Alternatively, a <a href="#">SummarizedExperiment</a> or <a href="#">SingleCellExperiment</a> object containing such a matrix.
...	For the generic, additional arguments to pass to specific methods. For the SummarizedExperiment and SingleCellExperiment methods, additional arguments to pass to the ANY method.

<code>size.factors.norm</code>	A numeric vector of size factors for normalization of <code>x</code> prior to PCA and distance calculations. If <code>NULL</code> , defaults to size factors derived from the library sizes of <code>x</code> . For the <code>SingleCellExperiment</code> method, the default values are taken from <code>sizeFactors(x)</code> , if they are available.
<code>size.factors.content</code>	A numeric vector of size factors for RNA content normalization of <code>x</code> prior to simulating doublets. This is orthogonal to the values in <code>size.factors.norm</code> , see <a href="#">Details</a> .
<code>k</code>	An integer scalar specifying the number of nearest neighbours to use to determine the bandwidth for density calculations.
<code>subset.row</code>	See <code>?"scran-gene-selection"</code> .
<code>niters</code>	An integer scalar specifying how many simulated doublets should be generated.
<code>block</code>	An integer scalar controlling the rate of doublet generation, to keep memory usage low.
<code>dims</code>	An integer scalar specifying the number of components to retain after the PCA.
<code>BNPARAM</code>	A <a href="#">BiocNeighborParam</a> object specifying the nearest neighbor algorithm. This should be an algorithm supported by <code>findNeighbors</code> .
<code>BSPARAM</code>	A <a href="#">BiocSingularParam</a> object specifying the algorithm to use for PCA, if <code>d</code> is not <code>NA</code> .
<code>BPPARAM</code>	A <a href="#">BiocParallelParam</a> object specifying whether the neighbour searches should be parallelized.
<code>assay.type</code>	A string specifying which assay values contain the count matrix.
<code>adjust</code>	Logical scalar indicating whether to adjust for differences in local density, see the vignette.

## Details

This function simulates doublets by adding the count vectors for two randomly chosen cells in `x`. For each original cell, we compute the density of neighboring simulated doublets and compare it to the density of neighboring original cells. Genuine doublets should have a high density of simulated doublets relative to the density of its neighbourhood. Thus, the doublet score for each cell is defined as the ratio of densities of simulated doublets to the density of the original cells.

Densities are calculated in low-dimensional space after a PCA on the log-normalized expression matrix of `x`. Simulated doublets are projected into the low-dimensional space using the rotation vectors computed from the original cells. For each cell, the density of simulated doublets is computed for a hypersphere with radius set to the median distance to the `k` nearest neighbour. This is normalized by `niters`, `k` and the total number of cells in `x` to yield the final score.

The two size factor arguments have different roles:

- `size.factors.norm` contains the size factors to be used for normalization prior to PCA and distance calculations. This defaults to the values returned by `librarySizeFactors` but can be explicitly set to ensure that the low-dimensional space is consistent with that in the rest of the analysis.
- `size.factors.content` is much more important, and represents the size factors that preserve RNA content differences. This is usually computed from spike-in RNA and ensures that the simulated doublets have the correct ratio of contributions from the original cells.

It is possible to set both of these arguments as they are orthogonal to each other. Setting `size.factors.content` will not affect the calculation of log-normalized expression values from `x`. Conversely, setting `size.factors.norm` will not affect the ratio in which cells are added together when simulating doublets.

### Value

A numeric vector of doublet scores for each cell in `x`.

### Author(s)

Aaron Lun

### References

Lun ATL (2018). Detecting doublet cells with *scran*. [https://l1la.github.io/SingleCellThoughts/software/doublet\\_detection/bycell.html](https://l1la.github.io/SingleCellThoughts/software/doublet_detection/bycell.html)

### See Also

[findDoubletClusters](#), to detect doublet clusters.

[scDblFinder](#), which uses a hybrid approach involving simulation and overclustering.

More detail on the mathematical background of this function is provided in the corresponding vignette at `vignette("computeDoubletDensity", package="scDblFinder")`.

### Examples

```
# Mocking up an example.
set.seed(100)
ngenes <- 1000
mu1 <- 2^rnorm(ngenes)
mu2 <- 2^rnorm(ngenes)
mu3 <- 2^rnorm(ngenes)
mu4 <- 2^rnorm(ngenes)

counts.1 <- matrix(rpois(ngenes*100, mu1), nrow=ngenes) # Pure type 1
counts.2 <- matrix(rpois(ngenes*100, mu2), nrow=ngenes) # Pure type 2
counts.3 <- matrix(rpois(ngenes*100, mu3), nrow=ngenes) # Pure type 3
counts.4 <- matrix(rpois(ngenes*100, mu4), nrow=ngenes) # Pure type 4
counts.m <- matrix(rpois(ngenes*20, mu1+mu2), nrow=ngenes) # Doublets (1 & 2)

counts <- cbind(counts.1, counts.2, counts.3, counts.4, counts.m)
clusters <- rep(1:5, c(rep(100, 4), ncol(counts.m)))

# Find potential doublets.
scores <- computeDoubletDensity(counts)
boxplot(split(log10(scores), clusters))
```

---

createDoublets      *createDoublets*

---

### Description

Creates artificial doublet cells by combining given pairs of cells

### Usage

```
createDoublets(
  x,
  dbl.idx,
  clusters = NULL,
  resamp = 0.5,
  halfSize = 0.5,
  adjustSize = FALSE,
  prefix = "dbl."
)
```

### Arguments

<code>x</code>	A count matrix of real cells
<code>dbl.idx</code>	A matrix or data.frame with pairs of cell indexes stored in the first two columns.
<code>clusters</code>	An optional vector of cluster labels (for each column of ‘x’)
<code>resamp</code>	Logical; whether to resample the doublets using the poisson distribution. Alternatively, if a proportion between 0 and 1, the proportion of doublets to resample.
<code>halfSize</code>	Logical; whether to half the library size of doublets (instead of just summing up the cells). Alternatively, a number between 0 and 1 can be given, determining the proportion of the doublets for which to perform the size adjustment. Ignored if not resampling.
<code>adjustSize</code>	Logical; whether to adjust the size of the doublets using the median sizes per cluster of the originating cells. Requires ‘clusters’ to be given. Alternatively to a logical value, a number between 0 and 1 can be given, determining the proportion of the doublets for which to perform the size adjustment.
<code>prefix</code>	Prefix for the colnames generated.

### Value

A matrix of artificial doublets.

### Examples

```
sce <- mockDoubletSCE()
idx <- getCellPairs(sce$cluster, n=200)
art.dbls <- createDoublets(sce, idx)
```

---

cxds2

*cxds2*


---

### Description

Calculates a coexpression-based doublet score using the method developed by [Bais and Kostka 2020](#). This is the original implementation from the ‘scds’ package, but enabling scores to be calculated for all cells while the gene coexpression is based only on a subset (i.e. excluding known/artificial doublets).

### Usage

```
cxds2(x, whichDbls = c(), ntop = 500, binThresh = 0)
```

### Arguments

x	A matrix of counts, or a ‘SingleCellExperiment’ containing a ‘counts’
whichDbls	The columns of ‘x’ which are known doublets.
ntop	The number of top features to keep.
binThresh	The count threshold to be considered expressed.

### Value

A cxds score or, if ‘x’ is a ‘SingleCellExperiment’, ‘x’ with an added ‘cxds\_score’ colData column.

### Examples

```
sce <- mockDoubletSCE()
sce <- cxds2(sce)
```

---

data

*Comparison results*


---

### Description

Results of the comparison of doublet detection methods. Each element of the list is a dataset, itself a list with scores for the different methods as well as the demuxlet SNP-based doublet calls (‘demuxlet\_cls’).

### Value

a list.

---

doubletThresholding     *doubletThresholding*

---

### Description

Sets the doublet scores threshold; typically called by `scDblFinder`.

### Usage

```
doubletThresholding(d, dbr = 0.025, dbr.sd = 0.015, local = FALSE)
```

### Arguments

<code>d</code>	A data.frame of properties of real and artificial cells, as produced by <code>'scDblFinder(returnType="table")'</code> .
<code>dbr</code>	The expected (mean) doublet rate.
<code>dbr.sd</code>	The standard deviation of the doublet rate, representing the uncertainty in the estimate.
<code>local</code>	Logical; whether to use local calibration (experimental feature!)

### Value

A scalar indicating the decided threshold.

### Examples

```
sce <- mockDoubletSCE()
d <- scDblFinder(sce, verbose=FALSE, returnType="table")
th <- doubletThresholding(d, dbr=0.05)
th$th
```

---

fastcluster     *fastcluster*

---

### Description

Performs a fast two-step clustering: first clusters using k-means with a very large k, then uses louvain clustering of the k cluster averages and reports back the cluster labels.

### Usage

```
fastcluster(
  x,
  k = NULL,
  rname = "PCA",
  nstart = 3,
  iter.max = 20,
  ndims = NULL,
```



```

nfeatures = 1000,
returnType = c("clusters", "preclusters", "metacells", "graph"),
BPPARAM = SerialParam()
)

```

### Arguments

x	An object of class SCE
k	The number of k-means clusters to use in the primary step (should be much higher than the number of expected clusters). Defaults to 1/10th of the number of cells with a maximum of 3000.
rdname	The name of the dimensionality reduction to use.
nstart	Number of starts for k-means clustering
iter.max	Number of iterations for k-means clustering
ndims	Number of dimensions to use
nfeatures	Number of features to use (ignored if 'rdname' is given and the corresponding dimensional reduction exists in 'sce')
returnType	See return.
BPPARAM	'BiocParallel' BPPARAM for multithreading.

### Value

By default, a vector of cluster labels. If 'returnType='preclusters'', returns the k-means pre-clusters. If 'returnType='metacells'', returns the metacells aggregated by pre-clusters and the corresponding cell indexes. If 'returnType='graph'', returns the graph of (meta-)cells and the corresponding cell indexes.

### Examples

```

sce <- mockDoubletSCE()
sce$cluster <- fastcluster(sce)

```

---

findDoubletClusters    *Detect doublet clusters*

---

### Description

Identify potential clusters of doublet cells based on whether they have intermediate expression profiles, i.e., their profiles lie between two other "source" clusters.

### Usage

```

findDoubletClusters(x, ...)

## S4 method for signature 'ANY'
findDoubletClusters(
  x,
  clusters,
  subset.row = NULL,

```

```

    threshold = 0.05,
    get.all.pairs = FALSE,
    ...
)

## S4 method for signature 'SummarizedExperiment'
findDoubletClusters(x, ..., assay.type = "counts")

## S4 method for signature 'SingleCellExperiment'
findDoubletClusters(x, clusters = collLabels(x, onAbsence = "error"), ...)

```

## Arguments

x	A numeric matrix-like object of count values, where each column corresponds to a cell and each row corresponds to an endogenous gene. Alternatively, a <a href="#">SummarizedExperiment</a> or <a href="#">SingleCellExperiment</a> object containing such a matrix.
...	For the generic, additional arguments to pass to specific methods. For the ANY method, additional arguments to pass to <a href="#">findMarkers</a> . For the SummarizedExperiment method, additional arguments to pass to the ANY method. For the SingleCellExperiment method, additional arguments to pass to the SummarizedExperiment method.
clusters	A vector of length equal to <code>ncol(x)</code> , containing cluster identities for all cells. If x is a <a href="#">SingleCellExperiment</a> , this is taken from <a href="#">collLabels(x)</a> by default.
subset.row	See <a href="#">?"scran-gene-selection"</a> .
threshold	A numeric scalar specifying the FDR threshold with which to identify significant genes.
get.all.pairs	Logical scalar indicating whether statistics for all possible source pairings should be returned.
assay.type	A string specifying which assay values to use, e.g., "counts" or "logcounts".

## Details

This function detects clusters of doublet cells in a manner similar to the method used by Bach et al. (2017). For each “query” cluster, we examine all possible pairs of “source” clusters, hypothesizing that the query consists of doublets formed from the two sources. If so, gene expression in the query cluster should be strictly intermediate between the two sources after library size normalization.

We apply pairwise t-tests to the normalized log-expression profiles to reject this null hypothesis. This is done by identifying genes that are consistently up- or down-regulated in the query compared to *both* sources. We count the number of genes that reject the null hypothesis at the specified FDR threshold. For each query cluster, the most likely pair of source clusters is that which minimizes the number of significant genes.

Potential doublet clusters are identified using the following characteristics, in order of importance:

- Low number of significant genes (i.e., `num.de`). Ideally, `median.de` is also high to indicate that the absence of strong DE is not due to a lack of power.
- A reasonable proportion of cells in the cluster, i.e., `prop`. This requires some expectation of the doublet rate in the experimental protocol.

- Library sizes of the source clusters that are below that of the query cluster, i.e., `lib.size*` values below unity. This assumes that the doublet cluster will contain more RNA and have more counts than either of the two source clusters.

For each query cluster, the function will only report the pair of source clusters with the lowest `num.de`. Setting `get.all.pairs=TRUE` will retrieve statistics for all pairs of potential source clusters. This can be helpful for diagnostics to identify relationships between specific clusters.

The reported `p.value` is of little use in a statistical sense, and is only provided for inspection. Technically, it could be treated as the Simes combined p-value against the doublet hypothesis for the query cluster. However, this does not account for the multiple testing across all pairs of clusters for each chosen cluster, especially as we are choosing the pair that is most concordant with the doublet null hypothesis.

We use library size normalization (via [librarySizeFactors](#)) even if existing size factors are present. This is because intermediate expression of the doublet cluster is not guaranteed for arbitrary size factors. For example, expression in the doublet cluster will be higher than that in the source clusters if normalization was performed with spike-in size factors.

### Value

A [DataFrame](#) containing one row per query cluster with the following fields:

`source1`: String specifying the identity of the first source cluster.

`source2`: String specifying the identity of the second source cluster.

`num.de`: Integer, number of genes that are significantly non-intermediate in the query cluster compared to the two putative source clusters.

`median.de`: Integer, median number of genes that are significantly non-intermediate in the query cluster across all possible source cluster pairings.

`best`: String specifying the identity of the top gene with the lowest p-value against the doublet hypothesis for this combination of query and source clusters.

`p.value`: Numeric, containing the adjusted p-value for the best gene.

`lib.size1`: Numeric, ratio of the median library sizes for the first source cluster to the query cluster.

`lib.size2`: Numeric, ratio of the median library sizes for the second source cluster to the query cluster.

`prop`: Numeric, proportion of cells in the query cluster.

`all.pairs`: A [SimpleList](#) object containing the above statistics for every pair of potential source clusters, if `get.all.pairs=TRUE`.

Each row is named according to its query cluster.

### Author(s)

Aaron Lun

### References

Bach K, Pensa S, Grzelak M, Hadfield J, Adams DJ, Marioni JC and Khaled WT (2017). Differentiation dynamics of mammary epithelial cells revealed by single-cell RNA sequencing. *Nat Commun.* 8, 1:2128.

**See Also**

[findMarkers](#), to detect DE genes between clusters.

**Examples**

```
# Mocking up an example.
library(SingleCellExperiment)
sce <- mockDoubletSCE()

# Compute doublet-ness of each cluster:
dbl <- findDoubletClusters(counts(sce), sce$cluster)
dbl

# Narrow this down to clusters with very low 'N':
library(scuttle)
isOutlier(dbl$num.de, log=TRUE, type="lower")

# Get help from "lib.size" below 1.
dbl$lib.size1 < 1 & dbl$lib.size2 < 1
```

---

getArtificialDoublets *getArtificialDoublets*

---

**Description**

Create expression profiles of random artificial doublets.

**Usage**

```
getArtificialDoublets(
  x,
  n = 3000,
  clusters = NULL,
  resamp = 0.5,
  halfSize = 0.5,
  adjustSize = 0.2,
  propRandom = 0.1,
  selMode = c("proportional", "uniform", "sqrt"),
  n.meta.cells = 2,
  meta.triplets = TRUE
)
```

**Arguments**

x	A count matrix, with features as rows and cells as columns.
n	The approximate number of doublet to generate (default 3000).
clusters	The optional clusters labels to use to build cross-cluster doublets.
resamp	Logical; whether to resample the doublets using the poisson distribution. Alternatively, if a proportion between 0 and 1, the proportion of doublets to resample.

halfSize	Logical; whether to half the library size of doublets (instead of just summing up the cells). Alternatively, a number between 0 and 1 can be given, determining the proportion of the doublets for which to perform the size adjustment.
adjustSize	Logical; whether to adjust the size of the doublets using the ratio between each cluster's median library size. Alternatively, a number between 0 and 1 can be given, determining the proportion of the doublets for which to perform the size adjustment.
propRandom	The proportion of the created doublets that are fully random (default 0.1); the rest will be doublets created across clusters. Ignored if 'clusters' is NULL.
selMode	The cell pair selection mode for inter-cluster doublet generation, either 'uniform' (same number of doublets for each combination), 'proportional' (proportion expected from the clusters' prevalences), or 'sqrt' (roughly the square root of the expected proportion).
n.meta.cells	The number of meta-cell per cluster to create. If given, additional doublets will be created from cluster meta-cells. Ignored if 'clusters' is missing.
meta.triplets	Logical; whether to create triplets from meta cells. Ignored if 'clusters' is missing.

**Value**

A list with two elements: 'counts' (the count matrix of the artificial doublets) and 'origins' the clusters from which each artificial doublets originated (NULL if 'clusters' is not given).

**Examples**

```
m <- t(sapply( seq(from=0, to=5, length.out=50),
              FUN=function(x) rpois(30,x) ) )
doublets <- getArtificialDoublets(m, 30)
```

---

getCellPairs

*getCellPairs*


---

**Description**

Given a vector of cluster labels, returns pairs of cross-cluster cells

**Usage**

```
getCellPairs(x, n = 1000, ...)
```

**Arguments**

x	A vector of cluster labels for each cell, or a list containing metacells and graph
n	The number of cell pairs to obtain
...	Further arguments, for instance the 'k' vector of precluster labels if 'x' is a metacell graph.

**Value**

A data.frame with the columns

**Examples**

```
# create random labels
x <- sample(head(LETTERS), 100, replace=TRUE)
getCellPairs(x, n=6)
```

---

```
getExpectedDoublets     getExpectedDoublets
```

---

**Description**

getExpectedDoublets

**Usage**

```
getExpectedDoublets(x, dbr = NULL, only.heterotypic = TRUE)
```

**Arguments**

x                    A vector of cluster labels for each cell  
dbr                    The expected doublet rate.  
only.heterotypic     Logical; whether to return expectations only for heterotypic doublets

**Value**

The expected number of doublets of each combination of clusters

**Examples**

```
# random cluster labels
c1 <- sample(head(LETTERS,4), size=2000, prob=c(.4,.2,.2,.2), replace=TRUE)
getExpectedDoublets(c1)
```

---

```
mockDoubletSCE             mockDoubletSCE
```

---

**Description**

Creates a mock random single-cell experiment object with doublets

**Usage**

```
mockDoubletSCE(
  ncells = c(200, 300),
  ngenes = 200,
  mus = NULL,
  dbl.rate = 0.1,
  only.heterotypic = TRUE
)
```

**Arguments**

ncells	A positive integer vector indicating the number of cells per cluster (min 2 clusters)
ngenes	The number of genes to simulate. Ignored if ‘mus’ is given.
mus	A list of cluster averages.
dbl.rate	The doublet rate
only.heterotypic	Whether to create only heterotypic doublets

**Value**

A SingleCellExperiment object, with the colData columns ‘type’ indicating whether the cell is a singlet or doublet, and ‘cluster’ indicating from which cluster (or cluster combination) it was simulated.

**Examples**

```
sce <- mockDoubletSCE()
```

---

plotDoubletMap	<i>plotDoubletMap</i>
----------------	-----------------------

---

**Description**

Plots a heatmap of observed versus expected doublets. Requires the ‘ComplexHeatmap’ package.

**Usage**

```
plotDoubletMap(
  sce,
  colorBy = "enrichment",
  labelBy = "observed",
  addSizes = TRUE,
  col = NULL,
  column_title = "Clusters",
  row_title = "Clusters",
  column_title_side = "bottom",
  na_col = "white",
  ...
)
```

**Arguments**

sce	A SingleCellExperiment object on which ‘scDbfFinder’ has been run.
colorBy	Determines the color mapping. Either "enrichment" (for log2-enrichment over expectation) or any column of ‘metadata(sce)\$scDbfFinder.stats’
labelBy	Determines the cell labels. Either "enrichment" (for log2-enrichment over expectation) or any column of ‘metadata(sce)\$scDbfFinder.stats’
addSizes	Logical; whether to add the sizes of clusters to labels

```

col          The colors scale to use (passed to 'ComplexHeatmap::Heatmap')
column_title passed to 'ComplexHeatmap::Heatmap'
row_title    passed to 'ComplexHeatmap::Heatmap'
column_title_side
              passed to 'ComplexHeatmap::Heatmap'
na_col       color for NA cells
...          passed to 'ComplexHeatmap::Heatmap'

```

**Value**

a Heatmap object

---

```

recoverDoublets      Recover intra-sample doublets

```

---

**Description**

Recover intra-sample doublets that are neighbors to known inter-sample doublets in a multiplexed experiment.

**Usage**

```

recoverDoublets(x, ...)

## S4 method for signature 'ANY'
recoverDoublets(
  x,
  doublets,
  samples,
  k = 50,
  transposed = FALSE,
  subset.row = NULL,
  BNPARAM = KmknnParam(),
  BPPARAM = SerialParam()
)

## S4 method for signature 'SummarizedExperiment'
recoverDoublets(x, ..., assay.type = "logcounts")

## S4 method for signature 'SingleCellExperiment'
recoverDoublets(x, ..., use.dimred = NULL)

```

**Arguments**

**x** A log-expression matrix for all cells (including doublets) in columns and genes in rows. If `transposed=TRUE`, this should be a matrix of low-dimensional coordinates where each row corresponds to a cell. Alternatively, a [SummarizedExperiment](#) or [SingleCellExperiment](#) containing (i) a log-expression matrix in the `assays` as specified by `assay.type`, or (ii) a matrix of reduced dimensions in the `reducedDims` as specified by `use.dimred`.



...	For the generic, additional arguments to pass to specific methods. For the SummarizedExperiment method, additional arguments to pass to the ANY method. For the SingleCellExperiment method, additional arguments to pass to the SummarizedExperiment method.
doublets	A logical, integer or character vector specifying which cells in <code>x</code> are known (inter-sample) doublets.
samples	A numeric vector containing the relative proportions of cells from each sample, used to determine how many cells are to be considered as intra-sample doublets.
k	Integer scalar specifying the number of nearest neighbors to use for computing the local doublet proportions.
transposed	Logical scalar indicating whether <code>x</code> is transposed, i.e., cells in the rows.
subset.row	A logical, integer or character vector specifying the genes to use for the neighbor search. Only used when <code>transposed=FALSE</code> .
BNPARAM	A <a href="#">BiocNeighborParam</a> object specifying the algorithm to use for the nearest neighbor search.
BPPARAM	A <a href="#">BiocParallelParam</a> object specifying the parallelization to use for the nearest neighbor search.
assay.type	A string specifying which assay values contain the log-expression matrix.
use.dimred	A string specifying whether existing values in <code>reducedDims(x)</code> should be used.

## Details

In multiplexed single-cell experiments, we can detect doublets as libraries with labels for multiple samples. However, this approach fails to identify doublets consisting of two cells with the same label. Such cells may be problematic if they are still sufficiently abundant to drive formation of spurious clusters.

This function identifies intra-sample doublets based on the similarity in expression profiles to known inter-sample doublets. For each cell, we compute the proportion of the `k` neighbors that are known doublets. Of the “unmarked” cells that are not known doublets, those with top `X` largest proportions are considered to be intra-sample doublets. We use `samples` to obtain a reasonable estimate for `X`, see the vignette for details.

A larger value of `k` provides more stable estimates of the doublet proportion in each cell. However, this comes at the cost of assuming that each cell actually has `k` neighboring cells of the same state. For example, if a doublet cluster has fewer than `k` members, its doublet proportions will be “diluted” by inclusion of unmarked cells in the next-closest cluster.

## Value

A [DataFrame](#) containing one row per cell and the following fields:

- `proportion`, a numeric field containing the proportion of neighbors that are doublets.
- `known`, a logical field indicating whether this cell is a known inter-sample doublet.
- `predicted`, a logical field indicating whether this cell is a predicted intra-sample doublet.

The `metadata` contains `intra`, a numeric scalar containing the expected number of intra-sample doublets.

**Author(s)**

Aaron Lun

**See Also**

[doubletCells](#) and [doubletCluster](#), for alternative methods of doublet detection when no prior doublet information is available.

hashedDrops from the **DropletUtils** package, to identify doublets from cell hashing experiments.

More detail on the mathematical background of this function is provided in the corresponding vignette at `vignette("recoverDoublets", package="scDblFinder")`.

**Examples**

```
# Mocking up an example.
set.seed(100)
ngenes <- 1000
mu1 <- 2^rnorm(ngenes, sd=2)
mu2 <- 2^rnorm(ngenes, sd=2)

counts.1 <- matrix(rpois(ngenes*100, mu1), nrow=ngenes) # Pure type 1
counts.2 <- matrix(rpois(ngenes*100, mu2), nrow=ngenes) # Pure type 2
counts.m <- matrix(rpois(ngenes*20, mu1+mu2), nrow=ngenes) # Doublets (1 & 2)
all.counts <- cbind(counts.1, counts.2, counts.m)
lcounts <- scuttle::normalizeCounts(all.counts)

# Pretending that half of the doublets are known. Also pretending that
# the experiment involved two samples of equal size.
known <- 200 + seq_len(10)
out <- recoverDoublets(lcounts, doublets=known, k=10, samples=c(1, 1))
out
```

---

scDblFinder

*scDblFinder*


---

**Description**

Identification of heterotypic (or neotypic) doublets in single-cell RNAseq using cluster-based generation of artificial doublets.

**Usage**

```
scDblFinder(
  sce,
  clusters = NULL,
  samples = NULL,
  trajectoryMode = FALSE,
  artificialDoublets = NULL,
  knownDoublets = NULL,
  use.cxds = TRUE,
  nfeatures = 1000,
  dims = 20,
```

```

dbr = NULL,
dbr.sd = 0.015,
k = NULL,
includePCs = 1:5,
propRandom = 0.1,
propMarkers = 0,
returnType = c("sce", "table", "full"),
score = c("xgb", "xgb.local.optim", "weighted", "ratio"),
metric = "aucpr",
nrounds = 50,
max_depth = 5,
iter = 1,
threshold = TRUE,
verbose = is.null(samples),
BPPARAM = SerialParam(),
...
)

```

### Arguments

sce	A <a href="#">SummarizedExperiment-class</a> , <a href="#">SingleCellExperiment-class</a> , or array of counts.
clusters	The optional cluster assignments (if omitted, will run clustering). This is used to make doublets more efficiently. <code>clusters</code> should either be a vector of labels for each cell, or the name of a <code>colData</code> column of <code>sce</code> . Alternatively, if it is a single integer, will determine how many clusters to create (using k-means clustering). This options should be used when distinct subpopulations are not expected in the data (e.g. trajectories).
samples	A vector of the same length as <code>cells</code> (or the name of a column of <code>colData(x)</code> ), indicating to which sample each cell belongs. Here, a sample is understood as being processed independently. If omitted, doublets will be searched for with all cells together. If given, doublets will be searched for independently for each sample, which is preferable if they represent different captures. If your samples were multiplexed using cell hashes, what you want to give here are the different batches/wells (i.e. independent captures, since doublets cannot arise across them) rather than biological samples.
trajectoryMode	Logical; whether to generate doublets in trajectory mode (i.e. for datasets with gradients rather than separated subpopulations). See <code>vignette("scDblFinder")</code> for more details.
artificialDoublets	The approximate number of artificial doublets to create. If <code>NULL</code> , will be the maximum of the number of cells or $5 * nbClusters^2$ .
knownDoublets	An optional logical vector of known doublets (e.g. through cell barcodes), or the name of a <code>colData</code> column of <code>'sce'</code> containing that information.
use.cxds	Logical; whether to use the <code>'cxds'</code> scores in addition to information from artificial/known doublets as part of the predictors.
nfeatures	The number of top features to use (default 1000)
dims	The number of dimensions used.
dbr	The expected doublet rate. By default this is assumed to be 1% per thousand cells captured (so 4% among 4000 thousand cells), which is appropriate for 10x

	datasets. Corrections for homeotypic doublets will be performed on the given rate.
dbr.sd	The uncertainty range in the doublet rate, interpreted as a +/- around 'dbr'. During thresholding, deviation from the expected doublet rate will be calculated from these boundaries, and will be considered null within these boundaries.
k	Number of nearest neighbors (for KNN graph). If more than one value is given, the doublet density will be calculated at each k (and other values at the highest k), and all the information will be used by the classifier. If omitted, a reasonable set of values is used.
includePCs	The index of principal components to include in the predictors (e.g. 'include-PCs=1:2').
propRandom	The proportion of the artificial doublets which should be made of random cells (as opposed to inter-cluster combinations).
propMarkers	The proportion of features to select based on marker identification.
returnType	Either "sce" (default), "table" (to return the table of cell attributes including artificial doublets), or "full" (returns an SCE object containing both the real and artificial cells).
score	Score to use for final classification.
metric	Error metric to optimize during training (e.g. 'merror', 'logloss', 'auc', 'aucpr').
nrounds	Maximum rounds of boosting. If NULL, will be determined through cross-validation. When the training is based only on simulated doublets, we generally find lower limits to outperform cross-validation.
max_depth	Maximum depth of decision trees
iter	A positive integer indicating the number of scoring iterations (ignored if 'score' isn't based on classifiers). At each iteration, real cells that would be called as doublets are excluding from the training, and new scores are calculated. Recommended values are 1 or 2.
threshold	Logical; whether to threshold scores into binary doublet calls
verbose	Logical; whether to print messages and the thresholding plot.
BPPARAM	Used for multithreading when splitting by samples (i.e. when 'samples!=NULL'); otherwise passed to eventual PCA and K/SNN calculations.
...	further arguments passed to <a href="#">getArtificialDoublets</a> .

## Details

This function generates artificial doublets from clusters of real cells, evaluates their prevalence in the neighborhood of each cells, and uses this along with additional features to classify doublets. The approach is complementary to doublets identified via cell hashes and SNPs in multiplexed samples: the latter can identify doublets formed by cells of the same type from two samples, which are nearly undistinguishable from real cells transcriptionally, but cannot identify doublets made by cells of the same sample. See `vignette("scDbfFinder")` for more details on the method.

When multiple samples/captures are present, they should be specified using the `samples` argument. Although the classifier will be trained globally, thresholding and the more computationally-intensive steps will be performed separately for each sample (in parallel if `BPPARAM` is given).

When inter-sample doublets are available, they can be provided to 'scDbfFinder' through the `knownDoublets` argument to improve the identification of further doublets.

**Value**

The sce object with several additional colData columns, in particular 'scDblFinder.score' (the final score used) and 'scDblFinder.class' (whether the cell is called as 'doublet' or 'singlet'). See vignette("scDblFinder") for more details; for alternative return values, see the 'returnType' argument.

**Examples**

```
library(SingleCellExperiment)
sce <- mockDoubletSCE()
sce <- scDblFinder(sce, dbr=0.1)
table(truth=sce$type, call=sce$scDblFinder.class)
```

---

selFeatures	<i>selFeatures</i>
-------------	--------------------

---

**Description**

Selects features based on cluster-wise expression or marker detection, or a combination.

**Usage**

```
selFeatures(
  sce,
  clusters = NULL,
  nfeatures = 1000,
  propMarkers = 0,
  FDR.max = 0.05
)
```

**Arguments**

sce	A <a href="#">SummarizedExperiment-class</a> , <a href="#">SingleCellExperiment-class</a> with a 'counts' assay.
clusters	Optional cluster assignments. Should either be a vector of labels for each cell.
nfeatures	The number of features to select.
propMarkers	The proportion of features to select from markers (rather than on the basis of high expression). Ignored if 'clusters' isn't given.
FDR.max	The maximum marker binom FDR to be included in the selection. (see <a href="#">findMarkers</a> ).

**Value**

A vector of feature (i.e. row) names.

**Examples**

```
sce <- mockDoubletSCE()
selFeatures(sce, clusters=sce$cluster, nfeatures=5)
```

# Index

addDoublets, 2  
assays, 16

BiocNeighborParam, 4, 17  
BiocParallelParam, 4, 17  
BiocSingularParam, 4

collLabels, 10  
computeDoubletDensity, 3  
computeDoubletDensity, ANY-method  
    (computeDoubletDensity), 3  
computeDoubletDensity, SingleCellExperiment-method  
    (computeDoubletDensity), 3  
computeDoubletDensity, SummarizedExperiment-method  
    (computeDoubletDensity), 3  
createDoublets, 2, 6  
cxds2, 7

data, 7  
DataFrame, 11, 17  
doubletCells, 18  
doubletCluster, 18  
doubletsComparison (data), 7  
doubletThresholding, 8

fastcluster, 8  
findDoubletClusters, 5, 9  
findDoubletClusters, ANY-method  
    (findDoubletClusters), 9  
findDoubletClusters, SingleCellExperiment-method  
    (findDoubletClusters), 9  
findDoubletClusters, SummarizedExperiment-method  
    (findDoubletClusters), 9  
findMarkers, 10, 12, 21  
findNeighbors, 4

getArtificialDoublets, 12, 20  
getCellPairs, 13  
getExpectedDoublets, 14

librarySizeFactors, 4, 11

metadata, 17  
mockDoubletSCE, 14

plotDoubletMap, 15

recoverDoublets, 16  
recoverDoublets, ANY-method  
    (recoverDoublets), 16  
recoverDoublets, SingleCellExperiment-method  
    (recoverDoublets), 16  
recoverDoublets, SummarizedExperiment-method  
    (recoverDoublets), 16

reducedDims, 16, 17

scDblFinder, 5, 8, 18  
selfFeatures, 21  
SimpleList, 11  
SingleCellExperiment, 3, 10, 16  
sizeFactors, 4  
SummarizedExperiment, 3, 10, 16