Package 'DropletUtils'

October 16, 2018

Version 1.0.3

Date 2018-08-07

Title Utilities for Handling Single-Cell Droplet Data

Maintainer Aaron Lun <alun@wehi.edu.au>

Depends R (>= 3.5), BiocParallel, SingleCellExperiment

Imports S4Vectors, Matrix, methods, utils, stats, edgeR, Rcpp, rhdf5

Suggests testthat, beachmat, knitr, BiocStyle, rmarkdown, HDF5Array

biocViews SingleCell, Sequencing, RNASeq, GeneExpression, Transcriptomics, DataImport, Coverage

Description Provides a number of utility functions for handling single-cell (RNA-seq) data from droplet technologies such as 10X Genomics. This includes data loading, identification of cells from empty droplets, removal of barcode-swapped pseudo-cells, and downsampling of the count matrix.

License GPL-3

NeedsCompilation yes

VignetteBuilder knitr

LinkingTo Rcpp, beachmat, Rhdf5lib

SystemRequirements C++11

RoxygenNote 6.0.1

git_url https://git.bioconductor.org/packages/DropletUtils

git_branch RELEASE_3_7

git_last_commit 076f4fa

git_last_commit_date 2018-08-07

Date/Publication 2018-10-15

Author Aaron Lun [aut, cre],
Jonathan Griffiths [ctb],
Davis McCarthy [ctb]

R topics documented:

`barcodeRanks` *Calculate barcode ranks*

Description

Compute barcode rank statistics and identifry the knee and inflection points on the total count curve.

Usage

```
barcodeRanks(m, lower=100, fit.bounds=NULL, df=20, ...)
```

Arguments

`m`	A real sparse matrix object, either a dgTMatrix or dgCMatrix. Columns represent barcoded droplets, rows represent cells.
`lower`	A numeric scalar specifying the lower bound on the total UMI count, at or below which all barcodes are assumed to correspond to empty droplets.
`fit.bounds`	A numeric vector of length 2, specifying the lower and upper bouunds on the total UMI count for spline fitting.
`df, ...`	Further arguments to pass to `smooth.spline`.

Details

Analyses of droplet-based scRNA-seq data often show a plot of the log-total count against the log-rank of each barcode, where the highest ranks have the largest totals. This is equivalent to a transposed empirical cumulative density plot with log-transformed axes, which focuses on the barcodes with the largest counts. The `barcodeRanks` function will compute these ranks for all barcodes. Barcodes with the same total count receive the same average rank to avoid problems with discrete runs of the same total.

The function will also identify a number of interesting points on the curve for downstream use, namely the inflection and knee points. Both of these points correspond to a sharp transition between two components of the total count distribution, presumably reflecting the difference between empty droplets with little RNA and cell-containing droplets with much more RNA.

- The inflection point is computed as the point on the rank/total curve where the first derivative is minimized. The derivative is computed directly from all points on the curve with total counts greater than `lower`. This avoids issues with erratic behaviour of the curve at lower totals.

- The knee point is defined as the point on the curve where the signed curvature is minimized. This requires calculation of the second derivative, which is much more sensitive to noise in the curve. To overcome this, a smooth spline is fitted to the log-total counts against the log-rank using the `smooth.spline` function. Derivatives are then calculated from the fitted spline using `predict`.

We supply a default setting of `df` to avoid overfitting the spline, which results in unstability in the higher derivatives (and thus the curvature). This and other arguments to `smooth.spline` can be tuned if the estimated knee point is not at an appropriate location. We also restrict the fit to lie within the bounds defined by `fit.bounds` to focus on the region containing the knee point. This allows us to obtain an accurate fit with low `df`, rather than attempting to model the entire curve.

If `fit.bounds` is not specified, the upper bound is automatically set to the inflection point, which should lie after the knee point. The lower bound is set to the point at which the first derivative is closest to zero, i.e., the "plateau" region before the knee point. Note that only points with total counts above `lower` will be considered, regardless of how `fit.bounds` is defined.

Value

A list with the following elements:

`rank`: A numeric vector of average ranks for each column of `m`.

`total`: A numeric vector of total counts for each column of `m`.

`fitted`: A numeric vector of fitted total counts from the spline for each column of `m`. This is `NA` for points with `x` outside of `fit.bounds`.

`knee`: A numeric scalar containing the total count at the knee point.

`inflection`: A numeric scalar containing the total count at the inflection point.

Author(s)

Aaron Lun

See Also

`emptyDrops`

Examples

```
# Mocking up some data:
set.seed(2000)
my.counts <- DropletUtils:::simCounts()

# Computing barcode rank statistics:
br.out <- barcodeRanks(my.counts)
names(br.out)

# Making a plot.
plot(br.out$rank, br.out$total, log="xy", xlab="Rank", ylab="Total")
o <- order(br.out$rank)
lines(br.out$rank[o], br.out$fitted[o], col="red")
abline(h=br.out$knee, col="dodgerblue", lty=2)
abline(h=br.out$inflection, col="forestgreen", lty=2)
legend("bottomleft", lty=2, col=c("dodgerblue", "forestgreen"),
    legend=c("knee", "inflection"))
```

`defaultDrops` *Call cells from number of UMIs*

Description

Call cells according to the number of UMIs associated with each barcode, as implemented in CellRanger.

Usage

```
defaultDrops(m, expected=3000, upper.quant=0.99, lower.prop=0.1)
```

Arguments

`m`	A real sparse matrix object, either a dgTMatrix or dgCMatrix. Columns represent barcoded droplets, rows represent cells. The matrix should correspond to an individual sample.
`expected`	A numeric scalar specifying the expected number of cells in this sample, as specified in the call to CellRanger.
`upper.quant`	A numeric scalar between 0 and 1 specifying the quantile of the top `expected` barcodes to consider for the first step of the algorithm
`lower.prop`	A numeric scalar between 0 and 1 specifying the fraction of molecules of the `upper.quant` quantile result that a barcode must exceed to be called as a cell

Details

The `defaultDrops` function will call cells based on library size similarly to the CellRanger software suite from 10X Genomics. Default arguments correspond to an exact reproduction of CellRanger's algorithm, where the number of expected cells was also left at CellRanger default value.

The method considers the `upper.quant` quantile of the top `expected` barcodes, ordered by decreasing number of UMIs, as a threshold. Any barcodes containing more molecules than `lower.prop` times this threshold is considered to be a cell, and is retained for further analysis.

This method may be vulnerable to calling very well-captured background RNA, or missing real cells that were poorly captured. See `?emptyDrops` for an alternative approach.

Value

`defaultDrops` will return a logical vector of length `ncol(m)`. Each element of the vector reports whether each column of `m` was called as a cell.

Author(s)

Jonathan Griffiths

References

10X Genomics (2017). Cell Ranger Algorithms Overview. https://support.10xgenomics.com/single-cell-gene-expression/software/pipelines/latest/algorithms/overview

See Also

emptyDrops

Examples

```
# Mocking up some data:
set.seed(0)
my.counts <- DropletUtils:::simCounts()

# Identify likely cell-containing droplets.
called <- defaultDrops(my.counts)
table(called)

# Get matrix of called cells.
cell.counts <- my.counts[, called]
```

---

downsampleMatrix *Downsample a count matrix*

Description

Downsample a count matrix to a desired proportion for each cell.

Usage

```
downsampleMatrix(x, prop, bycol=TRUE)
```

Arguments

x	A numeric matrix of counts.
prop	A numeric scalar or, if bycol=TRUE, a vector of length ncol(x). All values should lie in [0, 1] specifying the downsampling proportion for the matrix or for each cell.
bycol	A logical scalar indicating whether downsampling should be performed on a column-by-column basis.

Details

Given multiple batches of very different sequencing depths, it can be beneficial to downsample the deepest batches to match the coverage of the shallowest batches. This avoids differences in technical noise that can drive clustering by batch.

If bycol=TRUE, sampling without replacement is performed on the count vector for each cell. This yields a new count vector where the total is equal to prop times the original total count. Each count in the returned matrix is guaranteed to be smaller than the original value in x. Different proportions can be specified for different cells by setting prop to a vector.

If bycol=FALSE, downsampling without replacement is performed on the entire matrix. This yields a new matrix where the total count across all cells is equal to prop times the original total. The new total count for each cell may not be exactly equal to prop times the original value, which may or may not be more appropriate than bycol=TRUE for particular applications.

Technically, downsampling on the reads with downsampleReads is more appropriate as it recapitulates the effect of differences in sequencing depth per cell. However, in practice, the aim is to obtain

cells that have similar total counts across batches, for which downsampling on the UMI counts is a more direct approach.

Note that this function was originally implemented in the **scater** package as `downsampleCounts`.

Value

A numeric matrix of downsampled counts, of the same type as `x`.

Author(s)

Aaron Lun

See Also

`downsampleReads`

Examples

```
example(read10xCounts)
downsampled <- downsampleMatrix(counts(sce10x), prop = 0.5)
```

---

downsampleReads *Downsample reads in a 10X Genomics dataset*

Description

Generate a UMI count matrix after downsampling reads from the molecule information file produced by CellRanger for 10X Genomics data.

Usage

```
downsampleReads(sample, prop, barcode.length=NULL, bycol=FALSE)
```

Arguments

`sample`	A string containing the path to the molecule information HDF5 file.
`barcode.length`	An integer scalar specifying the length of the cell barcode, see `read10xMolInfo`.
`prop`	A numeric scalar or, if `bycol=TRUE`, a vector of length `ncol(x)`. All values should lie in [0, 1] specifying the downsampling proportion for the matrix or for each cell.
`bycol`	A logical scalar indicating whether downsampling should be performed on a column-by-column basis.

Details

This function downsamples the reads for each molecule by the specified `prop`, using the information in `sample`. It then constructs a UMI count matrix based on the molecules with non-zero read counts. The aim is to eliminate differences in technical noise that can drive clustering by batch, as described in `downsampleMatrix`.

Subsampling the reads with `downsampleReads` recapitulates the effect of differences in sequencing depth per cell. This provides an alternative to downsampling with the CellRanger `aggr` function or subsampling with the 10X Genomics R kit. Note that this differs from directly subsampling the UMI count matrix with `downsampleMatrix`.

If `bycol=TRUE`, sampling without replacement is performed on the reads for each cell. The total number of reads for each cell after downsampling is guaranteed to be `prop` times the original total. Different proportions can be specified for different cells by setting `prop` to a vector.

If `bycol=FALSE`, downsampling without replacement is performed on all reads from the entire dataset. The total number of reads for each cell after downsampling may not be exactly equal to `prop` times the original value. Note that this is the more natural approach and is the default, which differs from the default used in `downsampleMatrix`.

Value

A numeric sparse matrix containing the downsampled UMI counts for each gene (row) and barcode (column).

Author(s)

Aaron Lun

See Also

`downsampleMatrix`, `read10xMolInfo`

Examples

```
# Mocking up some 10X HDF5-formatted data.
out <- DropletUtils:::sim10xMolInfo(tempfile(), nsamples=1)

# Downsampling by the reads.
downsampleReads(out, barcode.length=4, prop=0.5)
```

---

emptyDrops *Identify empty droplets*

Description

Distinguish between droplets containing cells and ambient RNA in a droplet-based single-cell RNA sequencing experiment.

Usage

```
testEmptyDrops(m, lower=100, niters=10000, test.ambient=FALSE,
    ignore=NULL, BPPARAM=SerialParam())

emptyDrops(m, lower=100, retain=NULL, barcode.args=list(), ...)
```

Arguments

m	A numeric matrix object, usually a dgTMatrix or dgCMatrix. Columns represent barcoded droplets, rows represent genes.
lower	A numeric scalar specifying the lower bound on the total UMI count, at or below which all barcodes are assumed to correspond to empty droplets.
niters	An integer scalar specifying the number of iterations to use for the Monte Carlo p-value calculations.
test.ambient	A logical scalar indicating whether results should be returned for barcodes with totals less than or equal to lower.
ignore	A numeric scalar specifying the lower bound on the total UMI count, at or below which barcodes will be ignored (see Details for how this differs from lower).
BPPARAM	A BiocParallelParam object indicating whether parallelization should be used to compute p-values.
retain	A numeric scalar specifying the threshold for the total UMI count above which all barcodes are assumed to contain cells.
barcode.args	Further arguments to pass to barcodeRanks.
...	Further arguments to pass to testEmptyDrops.

Value

testEmptyDrops will return a DataFrame with the following components:

Total: Integer, the total UMI count for each barcode.

LogProb: Numeric, the log-probability of observing the barcode's count vector under the null model.

PValue: Numeric, the Monte Carlo p-value against the null model.

Limited: Logical, indicating whether a lower p-value could be obtained by increasing npts.

For barcodes with counts below lower, NA values are returned for all fields if test.ambient=FALSE. This is to ensure that the number of rows in the output DataFrame is identical to ncol(m).

emptyDrops will return a DataFrame like testEmptyDrops, with an additional FDR field.

Details about testEmptyDrops

The testEmptyDrops function will obtain an estimate of the composition of the ambient pool of RNA based on the barcodes with total UMI counts below lower. This assumes that a cell-containing droplet would generally have higher total counts than empty droplets containing RNA from the ambient pool. Counts for the low-count barcodes are pooled together, and an estimate of the proportion vector for the ambient pool is calculated using goodTuringProportions. The count vector for each barcode above lower is then tested for a significant deviation from these proportions.

The null hypothesis is that transcript molecules are included into droplets by multinomial sampling from the ambient profile. For each barcode, the probability of obtaining its count vector based on the null model is computed. Then, niters count vectors are simulated from the null model. The proportion of simulated vectors with probabilities lower than the observed multinomial probability for that barcode is used to calculate the p-value. We use this Monte Carlo approach as an exact multinomial p-value is difficult to calculate.

The ignore argument can also be set to ignore barcodes with total counts less than or equal to ignore. This differs from the lower argument in that the ignored barcodes are not necessarily used to compute the ambient profile. Users can interpret ignore as the minimum total count required for a barcode to be considered as a potential cell. In contrast, lower is the maximum total count below which all barcodes are assumed to be empty droplets.

Details about `emptyDrops`

The `emptyDrops` function combines the results of `testEmptyDrops` with `barcodeRanks` to identify droplets that are likely to contain cells. Specifically, the total count `K` at the knee point is determined, and barcodes that contain more than `K` total counts are always retained. This ensures that cells with profiles that are very similar to the ambient pool are not inadvertently discarded. If `retain` is specified, this is used instead of `K`, which may be useful if the knee point was not correctly identified in complex log-rank curves. Users can set `retain=Inf` to disable automatic retention of barcodes with large totals.

The Benjamini-Hochberg correction is also applied to the Monte Carlo p-values to correct for multiple testing. Cells can then be defined by taking all barcodes with significantly non-ambient profiles, e.g., at a false discovery rate of 1%. All barcodes with total counts above `K` (or `retain`) are assigned p-values of zero *during correction*, reflecting our assumption that they are true positives. This ensures that their Monte Carlo p-values do not affect the correction of other genes, and also means that they will have FDR values of zero. Nonetheless, their original Monte Carlo p-values are still reported in the output.

Author(s)

Aaron Lun

References

Lun A, Riesenfeld S, Andrews T, Dao TP, Gomes T, participants in the 1st Human Cell Atlas Jamboree, Marioni JC (2018). Distinguishing cells from empty droplets in droplet-based single-cell RNA sequencing data. *biorXiv*.

Phipson B, Smyth GK (2010). Permutation P-values should never be zero: calculating exact P-values when permutations are randomly drawn. *Stat. Appl. Genet. Mol. Biol.* 9:Article 39.

See Also

`barcodeRanks`, `defaultDrops`

Examples

```
# Mocking up some data:
set.seed(0)
my.counts <- DropletUtils:::simCounts()

# Identify likely cell-containing droplets.
out <- emptyDrops(my.counts)
out

is.cell <- out$FDR <= 0.01
sum(is.cell, na.rm=TRUE)

# Check if p-values are lower-bounded by 'npts'
# (increase 'npts' if any Limited==TRUE and Sig==FALSE)
table(Sig=is.cell, Limited=out$Limited)
```

makeCountMatrix *Make a count matrix*

Description

Construct a count matrix from per-molecule information, typically the cell and gene of origin.

Usage

```
makeCountMatrix(gene, cell, all.genes=NULL, all.cells=NULL, value=NULL)
```

Arguments

gene	An integer or character vector specifying the gene to which each molecule was assigned.
cell	An integer or character vector specifying the cell to which each molecule was assigned.
all.genes	A character vector containing the names of all genes in the dataset.
all.cells	A character vector containing the names of all cells in the dataset.
value	A numeric vector containing values for each molecule.

Details

Each element of the vectors gene, cell and (if specified) value contain information for a single transcript molecule. Each entry of the output matrix corresponds to a single gene and cell combination. If multiple molecules are present with the same combination, their values in value are summed together, and the sum is used as the entry of the output matrix.

If value=NULL, it will default to a vector of all 1's. Each entry of the output matrix represents the number of molecules with the corresponding combination, i.e., UMI counts. Users can pass other metrics such as the number of reads covering each molecule. This would yield a read count matrix rather than a UMI count matrix.

If all.genes is not specified, it is kept as NULL for integer gene. Otherwise, it is defined as the sorted unique values of character gene. The same occurs for cell and all.cells.

If gene is integer, its values should be positive and no greater than length(all.genes) if all.genes!=NULL. If gene is character, its values should be a subset of those in all.genes. The same requirements apply to cell and all.cells.

Value

A sparse matrix where rows are genes, columns are cells and entries are the sum of value for each gene/cell combination. Rows and columns are named if the gene or cell are character or if all.genes or all.cells are specified.

Author(s)

Aaron Lun

See Also

read10xMolInfo

Examples

```
nmolecules <- 100
gene.id <- sample(LETTERS, nmolecules, replace=TRUE)
cell.id <- sample(20, nmolecules, replace=TRUE)
makeCountMatrix(gene.id, cell.id)
```

`read10xCounts` *Load in data from 10x experiment*

Description

Creates a SingleCellExperiment from the CellRanger output directories for 10X Genomics data.

Usage

```
read10xCounts(samples, col.names=FALSE, ...)
```

Arguments

`samples`	A character vector containing one or more directory names, each corresponding to a 10X sample. Each directory should contain the `"matrix.mtx"`, `"genes.tsv"` and `"barcodes.tsv"` files generated by CellRanger.
`col.names`	A logical scalar indicating whether the columns of the output object should be named with the cell barcodes.
`...`	Further arguments to pass to `read10xMatrix`.

Details

This function was originally developed from the `Read10X` function from the **Seurat** package. It was then taken from the `read10xResults` implementation in the **scater** package.

If `col.names=TRUE` and `length(sample)==1`, each column is named by the cell barcode. For multiple samples, the columns are unnamed to avoid problems with non-unique barcodes across samples.

Note that user-level manipulation of sparse matrices requires loading of the **Matrix** package. Otherwise, calculation of `rowSums`, `colSums`, etc. will result in errors.

Value

A SingleCellExperiment object with counts data stored as a sparse matrix. Matrices are combined by column if multiple `samples` were specified. Rows are named with the gene identifier (usually ENSEMBL).

Author(s)

Davis McCarthy, with modifications from Aaron Lun

References

Zheng GX, Terry JM, Belgrader P, and others (2017). Massively parallel digital transcriptional profiling of single cells. *Nat Commun* 8:14049.

10X Genomics (2017). Gene-Barcode Matrices. https://support.10xgenomics.com/single-cell-gene-expressi
software/pipelines/latest/output/matrices

See Also

write10xCounts

Examples

```
# Mocking up some 10X genomics output.
example(write10xCounts)

# Reading it in.
sce10x <- read10xCounts(tmpdir)

# Column names are dropped with multiple 'samples'.
sce10x2 <- read10xCounts(c(tmpdir, tmpdir))
```

---

read10xMatrix *Read in the 10x count matrix*

Description

Creates a sparse or HDF5-backed count matrix from the MatrixMarket file produced by CellRanger.

Usage

```
read10xMatrix(file, hdf5.out=FALSE, chunk.size)
```

Arguments

file	String containing the path to a MatrixMarket file, usually named "matrix.mtx".
hdf5.out	A logical scalar indicating whether a HDF5Matrix object should be produced.
chunk.size	An integer scalar specifying the chunk size when reading in records from file.

Details

When hdf5.out=FALSE, readMM is used directly. However, for very large 10x experiments with more than .Machine$integer.max non-zero entries, dgCMatrix may encounter integer overflows. In such cases, setting hdf5.out=TRUE will produce a HDF5Matrix object instead.

Value

A dgCMatrix object (or a HDF5Matrix object, if hdf5.out=TRUE) containing the counts for each gene (row) and cell barcode (column).

Author(s)

Aaron Lun

References

10X Genomics (2017). Gene-Barcode Matrices. https://support.10xgenomics.com/single-cell-gene-expressi
software/pipelines/latest/output/matrices

See Also

readMM, read10xCounts

Examples

```
# Mocking up some 10X genomics output.
example(write10xCounts)

mm.path <- file.path(tmpdir, "matrix.mtx")
X <- read10xMatrix(mm.path)
altX <- read10xMatrix(mm.path, chunk.size=10, hdf5.out=TRUE)
```

---

read10xMolInfo *Read the 10X molecule information file*

Description

Extract relevant fields from the molecule information HDF5 file, produced by CellRanger for 10X Genomics data.

Usage

```
read10xMolInfo(sample, barcode.length=NULL, keep.unmapped=FALSE)
```

Arguments

sample	A string containing the path to the molecule information HDF5 file.
barcode.length	An integer scalar specifying the length of the cell barcode.
keep.unmapped	A logical scalar indicating whether unmapped molecules should be reported.

Details

Molecules that were not assigned to any gene have gene set to length(genes)+1. By default, these are removed when keep.unmapped=FALSE.

The length of the cell barcode is automatically inferred if barcode.length=NULL. Currently, version 1 of the 10X chemistry uses 14 nt barcodes, while version 2 uses 16 nt barcodes.

Value

A list is returned containing two elements. The first element is named data and is a DataFrame where each row corresponds to a single transcript molecule. The fields are as follows:

barcode: Character, the cell barcode for each molecule.

umi: Integer, the processed UMI barcode in 2-bit encoding.

gem_group: Integer, the GEM group.

gene: Integer, the index of the gene to which the molecule was assigned. This refers to an entry in the genes vector, see below.

reads: Integer, the number of reads mapped to this molecule.

The second element of the list is named genes and is a character vector containing the names of all genes in the annotation. This contains the names of the various entries of gene for the individual molecules.

Author(s)

Aaron Lun, based on code by Jonathan Griffiths

References

Zheng GX, Terry JM, Belgrader P, and others (2017). Massively parallel digital transcriptional profiling of single cells. *Nat Commun* 8:14049.

10X Genomics (2017). Molecule info. https://support.10xgenomics.com/single-cell-gene-expression/software/pipelines/latest/output/molecule_info

See Also

makeCountMatrix

Examples

```
# Mocking up some 10X HDF5-formatted data.
out <- DropletUtils:::sim10xMolInfo(tempfile())

# Reading the resulting file.
read10xMolInfo(out)
```

---

swappedDrops *Clean barcode-swapped droplet data*

Description

Remove the effects of barcode swapping on droplet-based single-cell RNA-seq data, specifically 10X Genomics datasets.

Usage

```
swappedDrops(samples, barcode.length=NULL, get.swapped=FALSE,
    get.diagnostics=FALSE, min.frac=0.8)
```

Arguments

samples	A character vector containing paths to the molecule information HDF5 files, produced by CellRanger for 10X Genomics data.
barcode.length	An integer scalar specifying the length of the cell barcode, see read10xMolInfo.
get.swapped	A logical scalar indicating whether the UMI counts corresponding to swapped molecules should also be returned.
get.diagnostics	A logical scalar indicating whether to return the number of reads for each swapped molecule in each sample.
min.frac	A numeric scalar specifying the minimum fraction of reads required for a swapped molecule to be assigned to a sample.

Details

Barcode swapping on the Illumina sequencer occurs when multiplexed samples undergo PCR re-amplification on the flow cell by excess primer with different barcodes. This results in sequencing of the wrong barcode and molecules being assigned to incorrect samples after debarcoding. With droplet data, there is the opportunity to remove such effects based on the combination of gene, UMI and cell barcode for each observed transcript molecule. It is very unlikely that the same combination will arise from different molecules in multiple samples. Thus, observation of the same combination across multiple samples is indicative of barcode swapping.

For each potentially swapped molecule in 10X Genomics data, the number of reads corresponding to that molecule is extracted from the molecule information file. The fraction of reads in each sample is calculated, and the molecule is assigned to the sample with the largest fraction if it is greater than `min.frac`. This assumes that the swapping rate is low, so the true sample of origin for a molecule should contain the majority of the reads. Setting `min.frac=1` will effectively remove all molecules that appear in multiple samples. We do not recommend setting `min.frac` lower than 0.5.

Value

A list is returned with the `cleaned` element, itself a list of sparse matrices is returned. Each matrix corresponds to a sample and contains the UMI count for each gene (row) and barcode (column) after removing swapped molecules. All barcodes that were originally observed are reported as columns, though note that it is possible for some barcodes to contain no counts.

If `get.swapped=TRUE`, a `swapped` element is returned in the top-level list. This contains sample-specific sparse matrices of UMI counts corresponding to the swapped molecules. Adding the cleaned and swapped matrices for each sample should yield the total UMI count prior to removal of swapped molecules.

If `get.diagnostics=TRUE`, the top-level list will also contain an additional `diagnostics` matrix. Each entry of this matrix contains the number of reads for each molecule (row) in each sample (column). This can be useful for examining the level of swapping across samples on a molecule-by-molecule basis.

Author(s)

Jonathan Griffiths, with modifications by Aaron Lun

References

Griffiths JA, Lun ATL, Richard AC, Bach K, Marioni JC (2017). Detection and removal of barcode swapping in single-cell RNA-seq data. *biorXiv* 177048.

See Also

`read10xMolInfo`

Examples

```
# Mocking up some 10x HDF5-formatted data, with swapping.
curfiles <- DropletUtils:::sim10xMolInfo(tempfile(), nsamples=3)

# Obtaining count matrices with swapping removed.
out <- swappedDrops(curfiles)
lapply(out, dim)
```

```
out <- swappedDrops(curfiles, get.swapped=TRUE, get.diagnostics=TRUE)
names(out)
```

`write10xCounts` *Write count data in the 10x format*

Description

Create a directory containing the count matrix and cell/gene annotation from a sparse matrix of UMI counts, in the format produced by the CellRanger software suite.

Usage

```
write10xCounts(path, x, barcodes=colnames(x), gene.id=rownames(x),
    gene.symbol=gene.id, overwrite=FALSE)
```

Arguments

`x`	A sparse numeric matrix of UMI counts.
`path`	A string containing the path to the output directory.
`barcodes`	A character vector of cell barcodes, one per column of `x`.
`gene.id`	A character vector of gene identifiers, one per row of `x`.
`gene.symbol`	A character vector of gene symbols, one per row of `x`.
`overwrite`	A logical scalar specifying whether `path` should be overwritten if it already exists.

Value

A directory is produced at `path` containing the files `"matrix.mtx"`, `"barcodes.tsv"` and `"genes.tsv"`. A `TRUE` value is invisibly returned.

Author(s)

Aaron Lun

References

10X Genomics (2017). Gene-Barcode Matrices. https://support.10xgenomics.com/single-cell-gene-expressi
software/pipelines/latest/output/matrices

See Also

`read10xCounts`

Examples

```
# Mocking up some count data.
library(Matrix)
my.counts <- matrix(rpois(1000, lambda=5), ncol=10, nrow=100)
my.counts <- as(my.counts, "dgCMatrix")
cell.ids <- paste0("BARCODE-", seq_len(ncol(my.counts)))

ngenes <- nrow(my.counts)
gene.ids <- paste0("ENSG0000", seq_len(ngenes))
gene.symb <- paste0("GENE", seq_len(ngenes))

# Writing this to file:
tmpdir <- tempfile()
write10xCounts(tmpdir, my.counts, gene.id=gene.ids,
    gene.symbol=gene.symb, barcodes=cell.ids)
list.files(tmpdir)
```

Index