Title  Decontamination of single cell genomics data

Version  1.0.0

Description  This package contains implementation of DecontX (Yang et al. 2020), a decontamination algorithm for single-cell RNA-seq, and DecontPro (Yin et al. 2023), a decontamination algorithm for single cell protein expression data. DecontX is a novel Bayesian method to computationally estimate and remove RNA contamination in individual cells without empty droplet information. DecontPro is a Bayesian method that estimates the level of contamination from ambient and background sources in CITE-seq ADT dataset and decontaminate the dataset.

License  MIT + file LICENSE

Encoding  UTF-8

Roxygen  list(markdown = TRUE)

RoxygenNote  7.2.3

Suggests  BiocStyle, dplyr, knitr, rmarkdown, scran, SingleCellMultiModal, TENxPBMCData, testthat (>= 3.0.0)

Config/testthat/edition  3

Imports  celda, dbscan, DelayedArray, ggplot2, Matrix (>= 1.5.3), MCMCprecision, methods, patchwork, plyr, Rcpp (>= 0.12.0), RcppParallel (>= 5.0.1), reshape2, rstan (>= 2.18.1), rstantools (>= 2.2.0), S4Vectors, scater, Seurat, SingleCellExperiment, SummarizedExperiment, withr

Biarch  true

Depends  R (>= 4.3.0)

LinkingTo  BH (>= 1.66.0), Rcpp (>= 0.12.0), RcppEigen (>= 0.3.3.3.0), RcppParallel (>= 5.0.1), rstan (>= 2.18.1), StanHeaders (>= 2.18.0)

SystemRequirements  GNU make

VignetteBuilder  knitr

biocViews  SingleCell, Bayesian

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

git_branch  RELEASE_3_18
R topics documented:

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>decontX-package</td>
<td>2</td>
</tr>
<tr>
<td>.call_stan_vb</td>
<td>3</td>
</tr>
<tr>
<td>.process_stan_vb_out</td>
<td>3</td>
</tr>
<tr>
<td>decontPro</td>
<td>4</td>
</tr>
<tr>
<td>decontX</td>
<td>5</td>
</tr>
<tr>
<td>decontXcounts</td>
<td>8</td>
</tr>
<tr>
<td>fastNormProp</td>
<td>9</td>
</tr>
<tr>
<td>fastNormPropLog</td>
<td>9</td>
</tr>
<tr>
<td>fastNormPropSqrt</td>
<td>10</td>
</tr>
<tr>
<td>nonzero</td>
<td>10</td>
</tr>
<tr>
<td>plotBoxByCluster</td>
<td>11</td>
</tr>
<tr>
<td>plotDecontXContamination</td>
<td>12</td>
</tr>
<tr>
<td>plotDecontXMarkerExpression</td>
<td>13</td>
</tr>
<tr>
<td>plotDecontXMarkerPercentage</td>
<td>16</td>
</tr>
<tr>
<td>plotDensity</td>
<td>18</td>
</tr>
<tr>
<td>retrieveFeatureIndex</td>
<td>19</td>
</tr>
<tr>
<td>simulateContamination</td>
<td>20</td>
</tr>
</tbody>
</table>

Index 22

decOntX-package

The 'decontX' package.

Description

A DESCRIPTION OF THE PACKAGE

References

.call_stan_vb  

Call Stan variational bayes for inference

Description
Call Stan variational bayes for inference

Usage
.call_stan_vb(data, initial_condition)

Arguments
- data: A list of input data for Stan.
- initial_condition: Initial values for Stan params.

Value
Stan output

.process_stan_vb_out  

Process Stan output.

Description
Process Stan output.

Usage
.process_stan_vb_out(stan_vb_output, dat)

Arguments
- stan_vb_output: Stan variational bayes output
- dat: List of data input to stan vb

Value
Decomposed counts based on Stan estimate.
Decontaminate using decontPro

Usage

\[
\text{decontPro}(\text{object}, \text{cell\_type}, ...) \\
\]  

## S4 method for signature 'SingleCellExperiment'
\[
\text{decontPro}(\text{object}, \text{cell\_type}, \text{delta\_sd} = 2e-05, \text{background\_sd} = 2e-06, ...) \\
\]

## S4 method for signature 'Seurat'
\[
\text{decontPro}(\text{object}, \text{cell\_type}, \text{delta\_sd} = 2e-05, \text{background\_sd} = 2e-06, ...) \\
\]

## S4 method for signature 'ANY'
\[
\text{decontPro}(\text{object}, \text{cell\_type}, \text{delta\_sd} = 2e-05, \text{background\_sd} = 2e-06, ...) \\
\]

Arguments

- **object**: Data matrix NxM (feature x droplet).
- **cell_type**: 1xM vector of cell type. 1-based.
- **...**: Additional arguments for generics.
- **delta_sd**: Prior variance for ambient contamination level. Default to 2e-5.
- **background_sd**: Prior variance for background contamination level. Default to 2e-6.

Value

A list containing decontaminated counts, and estimated parameters.

Examples

# Simulated count matrix
\[
\text{counts} \leftarrow \text{matrix}(\text{sample}(1:10, 1000, \text{replace = TRUE}), \text{ncol = 10}) \\
\]

# Cell type indicator
\[
\text{k} \leftarrow \text{c}(1, 1, 2, 2, 2, 3, 3, 4, 4, 4) \\
\]

# Decontamination
\[
\text{out} \leftarrow \text{decontPro}(\text{counts}, \text{k}, 1e-2, 1e-2) \\
\]

# Decontaminated counts
\[
\text{decontaminated\_counts} \leftarrow \text{out}$\text{decontaminated\_counts} \\
\]
Description

Identifies contamination from factors such as ambient RNA in single cell genomic datasets.

Usage

decontX(x, ...)

## S4 method for signature 'SingleCellExperiment'
decontX(
  x,
  assayName = "counts",
  z = NULL,
  batch = NULL,
  background = NULL,
  bgAssayName = NULL,
  bgBatch = NULL,
  maxIter = 500,
  delta = c(10, 10),
  estimateDelta = TRUE,
  convergence = 0.001,
  iterLogLik = 10,
  varGenes = 5000,
  dbscanEps = 1,
  seed = 12345,
  logfile = NULL,
  verbose = TRUE
)

## S4 method for signature 'ANY'
decontX(
  x,
  z = NULL,
  batch = NULL,
  background = NULL,
  bgBatch = NULL,
  maxIter = 500,
  delta = c(10, 10),
  estimateDelta = TRUE,
  convergence = 0.001,
  iterLogLik = 10,
  varGenes = 5000,
  dbscanEps = 1,
  seed = 12345,
  logfile = NULL,
  verbose = TRUE
)
logfile = NULL,
verbose = TRUE
)

Arguments

x A numeric matrix of counts or a SingleCellExperiment with the matrix located in the assay slot under assayName. Cells in each batch will be subsetted and converted to a sparse matrix of class dgCMatrix from package Matrix before analysis. This object should only contain filtered cells after cell calling. Empty cell barcodes (low expression droplets before cell calling) are not needed to run DecontX.

... For the generic, further arguments to pass to each method.

assayName Character. Name of the assay to use if x is a SingleCellExperiment.

z Numeric or character vector. Cell cluster labels. If NULL, PCA will be used to reduce the dimensionality of the dataset initially, 'umap' from the 'uwot' package will be used to further reduce the dataset to 2 dimensions and the 'dbscan' function from the 'dbscan' package will be used to identify clusters of broad cell types. Default NULL.

batch Numeric or character vector. Batch labels for cells. If batch labels are supplied, DecontX is run on cells from each batch separately. Cells run in different channels or assays should be considered different batches. Default NULL.

background A numeric matrix of counts or a SingleCellExperiment with the matrix located in the assay slot under assayName. It should have the same data format as x except it contains the empty droplets instead of cells. When supplied, empirical distribution of transcripts from these empty droplets will be used as the contamination distribution. Default NULL.

bgAssayName Character. Name of the assay to use if background is a SingleCellExperiment. Default to same as assayName.

bgBatch Numeric or character vector. Batch labels for background. Its unique values should be the same as those in batch, such that each batch of cells have their corresponding batch of empty droplets as background, pointed by this parameter. Default to NULL.


da...ta Numeric Vector of length 2. Concentration parameters for the Dirichlet prior for the contamination in each cell. The first element is the prior for the native counts while the second element is the prior for the contamination counts. These essentially act as pseudocounts for the native and contamination in each cell. If estimateDelta = TRUE, this is only used to produce a random sample of proportions for an initial value of contamination in each cell. Then fit_dirichlet is used to update delta in each iteration. If estimateDelta = FALSE, then delta is fixed with these values for the entire inference procedure. Fixing delta and setting a high number in the second element will force decontX to be more aggressive and estimate higher levels of contamination at the expense of potentially removing native expression. Default c(10, 10).

estimateDelta Boolean. Whether to update delta at each iteration.
convergence Numeric. The EM algorithm will be stopped if the maximum difference in the contamination estimates between the previous and current iterations is less than this. Default 0.001.


varGenes Integer. The number of variable genes to use in dimensionality reduction before clustering. Variability is calculated using modelGeneVar function from the 'scran' package. Used only when z is not provided. Default 5000.

dbscanEps Numeric. The clustering resolution parameter used in 'dbscan' to estimate broad cell clusters. Used only when z is not provided. Default 1.

seed Integer. Passed to with_seed. For reproducibility, a default value of 12345 is used. If NULL, no calls to with_seed are made.

logfile Character. Messages will be redirected to a file named logfile. If NULL, messages will be printed to stdout. Default NULL.

verbose Logical. Whether to print log messages. Default TRUE.

Value

If x is a matrix-like object, a list will be returned with the following items:

decontXcounts: The decontaminated matrix. Values obtained from the variational inference procedure may be non-integer. However, integer counts can be obtained by rounding, e.g. round(decontXcounts).

contamination: Percentage of contamination in each cell.

estimates: List of estimated parameters for each batch. If z was not supplied, then the UMAP coordinates used to generated cell cluster labels will also be stored here.

z: Cell population/cluster labels used for analysis.

runParams: List of arguments used in the function call.

If x is a SingleCellExperiment, then the decontaminated counts will be stored as an assay and can be accessed with decontXcounts(x). The contamination values and cluster labels will also be stored in colData(x). estimates and runParams will be stored in metadata(x)$decontX. The UMAPs used to generated cell cluster labels will be stored in reducedDims slot in x.

Author(s)

Shiyi Yang, Yuan Yin, Joshua Campbell

Examples

# Generate matrix with contamination
s <- simulateContamination(seed = 12345)

library(SingleCellExperiment)
library(celda)
sce <- SingleCellExperiment(list(counts = s$observedCounts))
sce <- decontX(sce)

# Plot contamination on UMAP
plotDecontXContamination(sce)
# Plot decontX cluster labels
umap <- reducedDim(sce)
celda::plotDimReduceCluster(x = sce$decontX_clusters,  
dim1 = umap[, 1], dim2 = umap[, 2], )

# Plot percentage of marker genes detected  
# in each cell cluster before decontamination  
s$markers  
plotDecontXMarkerPercentage(sce, markers = s$markers, assayName = "counts")

# Plot percentage of marker genes detected  
# in each cell cluster after contamination  
plotDecontXMarkerPercentage(sce, markers = s$markers,  
    assayName = "decontXcounts")

# Plot percentage of marker genes detected in each cell  
# comparing original and decontaminated counts side-by-side  
plotDecontXMarkerPercentage(sce, markers = s$markers,  
    assayName = c("counts", "decontXcounts"))

# Plot raw counts of individual marker genes before  
# and after decontamination  
plotDecontXMarkerExpression(sce, unlist(s$markers))

decontXcounts  

Get or set decontaminated counts matrix

Description

Gets or sets the decontaminated counts matrix from a SingleCellExperiment object.

Usage

decontXcounts(object, ...)
decontXcounts(object, ...) <- value

## S4 method for signature 'SingleCellExperiment'
decontXcounts(object, ...)

## S4 replacement method for signature 'SingleCellExperiment'
decontXcounts(object, ...) <- value

Arguments

object A SingleCellExperiment object.
...
value A matrix to save as an assay called decontXcounts
Value

If getting, the assay from object with the name `decontXcounts` will be returned. If setting, a `SingleCellExperiment` object will be returned with `decontXcounts` listed in the assay slot.

See Also

`assay` and `assay<-`

---

### fastNormProp

**Fast normalization for numeric matrix**

**Description**

Fast normalization for numeric matrix

**Usage**

```r
fastNormProp(R_counts, R_alpha)
```

**Arguments**

- `R_counts`: An integer matrix
- `R_alpha`: A double value to be added to the matrix as a pseudocount

**Value**

A numeric matrix where the columns have been normalized to proportions

---

### fastNormPropLog

**Fast normalization for numeric matrix**

**Description**

Fast normalization for numeric matrix

**Usage**

```r
fastNormPropLog(R_counts, R_alpha)
```

**Arguments**

- `R_counts`: An integer matrix
- `R_alpha`: A double value to be added to the matrix as a pseudocount

**Value**

A numeric matrix where the columns have been normalized to proportions
**fastNormPropSqrt**
*Fast normalization for numeric matrix*

**Description**
Fast normalization for numeric matrix

**Usage**
`fastNormPropSqrt(R_counts, R_alpha)`

**Arguments**
- **R_counts**: An integer matrix
- **R_alpha**: A double value to be added to the matrix as a pseudocount

**Value**
A numeric matrix where the columns have been normalized to proportions

---

**nonzero**
*Get row and column indices of non-zero elements in the matrix*

**Description**
Get row and column indices of non-zero elements in the matrix

**Usage**
`nonzero(R_counts)`

**Arguments**
- **R_counts**: A matrix

**Value**
An integer matrix where each row is a row, column indices pair
### Description

Boxplot of features grouped by cell type.

### Usage

```r
plotBoxByCluster(
  counts,
  decontaminated_counts,
  cell_type,
  features,
  file = NULL
)
```

### Arguments

- **counts**: original count matrix of nADT x nDroplet.
- **decontaminated_counts**: decontaminated count matrix.
- **cell_type**: 1xnDroplet vector of cell_type.
- **features**: names of ADT to plot.
- **file**: file name to save plot into a pdf. If omit, return `ggplot` object.

### Value

Return a pdf file named `file` or a `ggplot` object.

### Examples

```r
# Simulate a dataset with 3 cells and 2 ADTs
counts <- matrix(c(60, 72, 52, 49, 89, 112),
                 nrow = 2,
                 dimnames = list(c('CD3', 'CD4'),
                                 c('CTGTTTACACCGCTAG',
                                   'CTCTACGGTGTGGCTC',
                                   'AGCAGCCAGGCTATT')))

decontaminated_counts <- matrix(c(58, 36, 26, 45, 88, 110),
                                nrow = 2,
                                dimnames = list(c('CD3', 'CD4'),
                                                c('CTGTTTACACCGCTAG',
                                                  'CTCTACGGTGTGGCTC',
                                                  'AGCAGCCAGGCTATT')))
```
plotDecontXContamination

Plots contamination on UMAP coordinates

Description

A scatter plot of the UMAP dimensions generated by DecontX with cells colored by the estimated percentage of contamination.

Usage

plotDecontXContamination(x, batch = NULL, colorScale = c("blue", "green", "yellow", "orange", "red"), size = 1)

Arguments

x Either a SingleCellExperiment with decontX results stored in metadata(x)$decontX or the result from running decontX on a count matrix.

batch Character. Batch of cells to plot. If NULL, then the first batch in the list will be selected. Default NULL.

colorScale Character vector. Contains the color spectrum to be passed to scale_colour_gradientn from package 'ggplot2'. Default c("blue","green","yellow","orange","red").


Value

Returns a ggplot object.

Author(s)

Shiyi Yang, Joshua Campbell

See Also

See decontX for a full example of how to estimate and plot contamination.
Examples

```r
# Generate matrix with contamination
s <- simulateContamination(seed = 12345)

library(SingleCellExperiment)
library(celda)
sce <- SingleCellExperiment(list(counts = s$observedCounts))
sce <- decontX(sce)

# Plot contamination on UMAP
plotDecontXContamination(sce)

# Plot decontX cluster labels
umap <- reducedDim(sce)
celda::plotDimReduceCluster(x = sce$decontX_clusters,
                           dim1 = umap[, 1], dim2 = umap[, 2], )

# Plot percentage of marker genes detected
# in each cell cluster before decontamination
s$markers
plotDecontXMarkerPercentage(sce, markers = s$markers, assayName = "counts")

# Plot percentage of marker genes detected
# in each cell cluster after contamination
plotDecontXMarkerPercentage(sce, markers = s$markers,
                           assayName = "decontXcounts")

# Plot percentage of marker genes detected in each cell
# comparing original and decontaminated counts side-by-side
plotDecontXMarkerPercentage(sce, markers = s$markers,
                           assayName = c("counts", "decontXcounts"))

# Plot raw counts of individual markers genes before
# and after decontamination
plotDecontXMarkerExpression(sce, unlist(s$markers))
```

---

**plotDecontXMarkerExpression**

Plots expression of marker genes before and after decontamination

**Description**

Generates a violin plot that shows the counts of marker genes in cells across specific clusters or cell types. Can be used to view the expression of marker genes in different cell types before and after decontamination with `decontX`.

**Usage**

```
plotDecontXMarkerExpression()
```
plotDecontXMarkerExpression

Arguments

x
Either a SingleCellExperiment or a matrix-like object of counts.

markers
Character Vector or List. A character vector or list of character vectors with the
names of the marker genes of interest.

groupClusters
List. A named list that allows cell clusters labels coded in z to be regrouped and
renamed on the fly. For example, list(Tcells=c(1, 2), Bcells=7) would
recode clusters 1 and 2 to "Tcells" and cluster 7 to "Bcells". Note that if this is
used, clusters in z not found in groupClusters will be excluded. Default NULL.

assayName
Character vector. Name(s) of the assay(s) to plot if x is a SingleCellExperiment.
If more than one assay is listed, then side-by-side violin plots will be generated.
Default c("counts", "decontXcounts").

z
Character, Integer, or Vector. Indicates the cluster labels for each cell. If x is a
SingleCellExperiment and z = NULL, then the cluster labels from decontX will
be retrieved from the colData of x (i.e. colData(x)$decontX_clusters). If z
is a single character or integer, then that column will be retrieved from colData
of x. (i.e. colData(x)[,z]). If x is a counts matrix, then z will need to be a
vector the same length as the number of columns in x that indicate the cluster to
which each cell belongs. Default NULL.

effectMatch
Boolean. Whether to only identify exact matches for the markers or to iden-
tify partial matches using grep. See retrieveFeatureIndex for more details.
Default TRUE.

by
Character. Where to search for the markers if x is a SingleCellExperiment. See
retrieveFeatureIndex for more details. If x is a matrix, then this must be set
to "rownames". Default "rownames".

log1p
Boolean. Whether to apply the function log1p to the data before plotting. This
function will add a pseudocount of 1 and then log transform the expression val-
ues. Default FALSE.

ncol
Integer. Number of columns to make in the plot. Default NULL.

plotDots
Boolean. If TRUE, the expression of features will be plotted as points in addition
to the violin curve. Default FALSE.

dotSize
Numeric. Size of points if plotDots = TRUE. Default 0.1.
plotDecontXMarkerExpression

Value

Returns a ggplot object.

Author(s)

Shiyi Yang, Joshua Campbell

See Also

See decontX for a full example of how to estimate and plot contamination.

Examples

# Generate matrix with contamination
s <- simulateContamination(seed = 12345)

library(SingleCellExperiment)
library(celda)
sce <- SingleCellExperiment(list(counts = s$observedCounts))
sce <- decontX(sce)

# Plot contamination on UMAP
plotDecontXContamination(sce)

# Plot decontX cluster labels
umap <- reducedDim(sce)
celda::plotDimReduceCluster(x = sce$decontX_clusters,
  dim1 = umap[, 1], dim2 = umap[, 2], )

# Plot percentage of marker genes detected
# in each cell cluster before decontamination
s$markers
plotDecontXMarkerPercentage(sce, markers = s$markers, assayName = "counts")

# Plot percentage of marker genes detected
# in each cell cluster after contamination
plotDecontXMarkerPercentage(sce, markers = s$markers,
  assayName = "decontXcounts")

# Plot percentage of marker genes detected in each cell
# comparing original and decontaminated counts side-by-side
plotDecontXMarkerPercentage(sce, markers = s$markers,
  assayName = c("counts", "decontXcounts"))

# Plot raw counts of individual marker genes before
# and after decontamination
plotDecontXMarkerExpression(sce, unlist(s$markers))
**plotDecontXMarkerPercentage**

*Plots percentage of cells cell types expressing markers*

**Description**

Generates a barplot that shows the percentage of cells within clusters or cell types that have detectable levels of given marker genes. Can be used to view the expression of marker genes in different cell types before and after decontamination with decontX.

**Usage**

```r
plotDecontXMarkerPercentage(
  x,
  markers,
  groupClusters = NULL,
  assayName = c("counts", "decontXcounts"),
  z = NULL,
  threshold = 1,
  exactMatch = TRUE,
  by = "rownames",
  ncol = round(sqrt(length(markers))),
  labelBars = TRUE,
  labelSize = 3
)
```

**Arguments**

- **x**: Either a SingleCellExperiment or a matrix-like object of counts.
- **markers**: List. A named list indicating the marker genes for each cell type of interest. Multiple markers can be supplied for each cell type. For example, `list(Tcell_Markers=c("CD3E", "CD3D"), Bcell_Markers=c("CD79A", "CD79B", "MS4A1"))` would specify markers for human T-cells and B-cells. A cell will be considered "positive" for a cell type if it has a count greater than `threshold` for at least one of the marker genes in the list.
- **groupClusters**: List. A named list that allows cell clusters labels coded in `z` to be regrouped and renamed on the fly. For example, `list(Tcells=c(1, 2), Bcells=7)` would recode clusters 1 and 2 to "Tcells" and cluster 7 to "Bcells". Note that if this is used, clusters in `z` not found in groupClusters will be excluded from the barplot. Default NULL.
- **assayName**: Character vector. Name(s) of the assay(s) to plot if `x` is a SingleCellExperiment. If more than one assay is listed, then side-by-side barplots will be generated. Default `c("counts", "decontXcounts")`.
- **z**: Character, Integer, or Vector. Indicates the cluster labels for each cell. If `x` is a SingleCellExperiment and `z = NULL`, then the cluster labels from decontX will be retrieved from the colData of `x` (i.e. `colData(x)$decontX_clusters`). If `z`
plotDecontXMarkerPercentage

is a single character or integer, then that column will be retrieved from colData of x. (i.e. colData(x)[,z]). If x is a counts matrix, then z will need to be a vector the same length as the number of columns in x that indicate the cluster to which each cell belongs. Default NULL.

threshold Numeric. Markers greater than or equal to this value will be considered detected in a cell. Default 1.

exactMatch Boolean. Whether to only identify exact matches for the markers or to identify partial matches using grep. See retrieveFeatureIndex for more details. Default TRUE.

by Character. Where to search for the markers if x is a SingleCellExperiment. See retrieveFeatureIndex for more details. If x is a matrix, then this must be set to "rownames". Default "rownames".

ncol Integer. Number of columns to make in the plot. Default round(sqrt(length(markers)).

labelBars Boolean. Whether to display percentages above each bar. Default TRUE.

labelSize Numeric. Size of the percentage labels in the barplot. Default 3.

Value

Returns a ggplot object.

Author(s)

Shiyi Yang, Joshua Campbell

See Also

See decontX for a full example of how to estimate and plot contamination.

Examples

# Generate matrix with contamination
s <- simulateContamination(seed = 12345)

library(SingleCellExperiment)
library(celda)
sce <- SingleCellExperiment(list(counts = s$observedCounts))
sce <- decontX(sce)

# Plot contamination on UMAP
plotDecontXContamination(sce)

# Plot decontX cluster labels
umap <- reducedDim(sce)
celda::plotDimReduceCluster(x = sce$decontX_clusters, 
dim1 = umap[, 1], dim2 = umap[, 2], )

# Plot percentage of marker genes detected
# in each cell cluster before decontamination
smarkers
plotDecontXMarkerPercentage(sce, markers = s$markers, assayName = "counts")

# Plot percentage of marker genes detected
# in each cell cluster after contamination
plotDecontXMarkerPercentage(sce, markers = s$markers,
                           assayName = "decontXcounts")

# Plot percentage of marker genes detected in each cell
# comparing original and decontaminated counts side-by-side
plotDecontXMarkerPercentage(sce, markers = s$markers,
                           assayName = c("counts", "decontXcounts"))

# Plot raw counts of individual markers genes before
# and after decontamination
plotDecontXMarkerExpression(sce, unlist(s$markers))

---

### plotDensity

Density of each ADT, raw counts overlapped with decontaminated counts

#### Description

Density of each ADT, raw counts overlapped with decontaminated counts

#### Usage

```r
plotDensity(counts, decontaminated_counts, features, file = NULL)
```

#### Arguments

- `counts`: original count matrix of nADT x nDroplet.
- `decontaminated_counts`: decontaminated count matrix.
- `features`: names of ADT to plot.
- `file`: file name to save plot into a pdf. If omit, return ggplot object.

#### Value

Return a pdf file named file or a ggplot object.

#### Examples

```r
# Simulate a dataset with 3 cells and 2 ADTs
counts <- matrix(c(60, 72, 52, 49, 89, 112),
                 nrow = 2,
                 dimnames = list(c("CD3", "CD4"),
                                c("CTGTTTACACCGCTAG",
                                 "CTCTACGGTGTGGCTC",
                                 "AGCAGCCAGGCTCATT")))
```
decontaminated_counts <- matrix(c(58, 36, 26, 45, 88, 110),
nrow = 2,
dimnames = list(c('CD3', 'CD4'),
               c('CTGTACACCGCTAG',
                 'CTCTACGGTGCTC',
                 'AGCAGCCAGGCTCATT')))

plotDensity(counts,
            decontaminated_counts,
            c('CD3', 'CD4'))
Author(s)

Yusuke Koga, Joshua Campbell

See Also

'retrieveFeatureInfo' from package 'scater' and link{regex} for how to use regular expressions when exactMatch = FALSE.

Examples

```r
counts <- matrix(sample(1:10, 20*10, replace = TRUE),
    nrow = 20, ncol = 10,
    dimnames = list(paste0("Gene_",1:20),
                     paste0("Cell_", 1:10)))
retrieveFeatureIndex(c("Gene_1", "Gene_5"), counts)
retrieveFeatureIndex(c("Gene_1", "Gene_5"), counts, exactMatch = FALSE)
```

---

**simulateContamination**

_Simulate contaminated count matrix_

**Description**

This function generates a list containing two count matrices – one for real expression, the other one for contamination, as well as other parameters used in the simulation which can be useful for running decontamination.

**Usage**

```r
simulateContamination(
    C = 300,
    G = 100,
    K = 3,
    NRange = c(500, 1000),
    beta = 0.1,
    delta = c(1, 10),
    numMarkers = 3,
    seed = 12345
)
```

**Arguments**

- **C** integer. Number of cells to be simulated. Default 300.
- **G** integer. Number of genes to be simulated. Default 100.
- **K** integer. Number of cell populations to be simulated. Default 3.
- **NRange** integer vector. A vector of length 2 that specifies the lower and upper bounds of the number of counts generated for each cell. Default c(500, 1000).
### simulateContamination

**beta**
Numeric. Concentration parameter for Phi. Default 0.1.

**delta**
Numeric or Numeric vector. Concentration parameter for Theta. If input as a single numeric value, symmetric values for beta distribution are specified; if input as a vector of length 2, the two values will be the shape1 and shape2 parameters of the beta distribution respectively. Default c(1, 5).

**numMarkers**
Integer. Number of markers for each cell population. Default 3.

**seed**
Integer. Passed to `with_seed`. For reproducibility, a default value of 12345 is used. If NULL, no calls to `with_seed` are made.

### Value
A list containing the `nativeMatirx` (real expression), `observedMatrix` (real expression + contamination), as well as other parameters used in the simulation.

### Author(s)
Shiyi Yang, Yuan Yin, Joshua Campbell

### Examples
```r
contaminationSim <- simulateContamination(K = 3, delta = c(1, 10))
```
Index

.call_stan_vb, 3
.process.stan_vb_out, 3

assay, 9

dbscan, 6, 7
decontPro, 4
decontPro,ANY-method (decontPro), 4
decontPro,Seurat-method (decontPro), 4
decontPro,SingleCellExperiment-method (decontPro), 4
decontX, 5, 12–17
decontX, ANY-method (decontX), 5
decontX,SingleCellExperiment-method (decontX), 5
decontX-package, 2
decontXcounts, 8
decontXcounts,SingleCellExperiment-method (decontXcounts), 8
decontXcounts<- (decontXcounts), 8
decontXcounts<-,SingleCellExperiment-method (decontXcounts), 8

fastNormProp, 9
fastNormPropLog, 9
fastNormPropSqrt, 10
fit_dirichlet, 6

grep, 14, 17, 19

Matrix, 6
modelGeneVar, 7

nonzero, 10

plotBoxByCluster, 11
plotDecontXContamination, 12
plotDecontXMarkerExpression, 13
plotDecontXMarkerPercentage, 16
plotDensity, 18

retrieveFeatureIndex, 14, 17, 19
retrieveFeatureInfo, 20

simulateContamination, 20
SingleCellExperiment, 6–9, 12, 14, 16, 17, 19
SummarizedExperiment, 19

umap, 6

with_seed, 7, 21