Package 'singleCellTK'

April 12, 2022

Type Package

Title Comprehensive and Interactive Analysis of Single Cell RNA-Seq Data

Version 2.4.0

Depends R (>= 4.0), SummarizedExperiment, SingleCellExperiment, DelayedArray, Biobase

Description Run common single cell analysis in the R console or directly through your browser. Includes many functions for import, quality control, normalization, batch correction, clustering, differential expression, and visualization..

License MIT + file LICENSE

Encoding UTF-8

biocViews SingleCell, GeneExpression, DifferentialExpression, Alignment, Clustering, ImmunoOncology

LazyData TRUE

Imports ape, batchelor, BiocParallel, celldex, colourpicker, colorspace, cowplot, cluster, ComplexHeatmap, data.table, DelayedMatrixStats, DESeq2, dplyr, DT, ExperimentHub, fields, ggplot2, ggplotify, ggrepel, ggtree, gridExtra, GSVA (>= 1.26.0), GSVAdata, igraph, KernSmooth, limma, MAST, Matrix, matrixStats, methods, msigdbr, multtest, plotly, RColorBrewer, ROCR, Rtsne, S4Vectors, scater, scMerge (>= 1.2.0), scran, Seurat (>= 3.1.3), shiny, shinyjs, SingleR, sva, reshape2, AnnotationDbi, shinyalert, circlize, enrichR, celda, shinycssloaders, DropletUtils, scds (>= 1.2.0), reticulate (>= 1.14), tools, tximport, fishpond, withr, GSEABase, R.utils, zinbwave, scRNAseq (>= 2.0.2), TENxPBMCData, yaml, rmarkdown, magrittr, scDblFinder, metap, VAM (>= 0.5.3), tibble, rlang, stats

RoxygenNote 7.1.1

Suggests testthat, Rsubread, BiocStyle, knitr, lintr, xtable, spelling, org.Mm.eg.db, stringr, kableExtra, shinythemes, shinyBS, shinyjqui, shinyWidgets, shinyFiles, BiocGenerics

VignetteBuilder knitr

URL https://www.camplab.net/sctk/

BugReports https://github.com/compbiomed/singleCellTK/issues

Language en-US

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

git_branch RELEASE_3_14

git_last_commit 91f98fc

git_last_commit_date 2021-10-27

Date/Publication 2022-04-12

Author Yichen Wang [aut, cre] (<https://orcid.org/0000-0003-4347-5199>),
Irzam Sarfraz [aut],
Rui Hong [aut],
Yusuke Koga [aut],
Salam Alabdullatif [aut],
David Jenkins [aut] (<https://orcid.org/0000-0002-7451-4288>),
Vidya Akavoor [aut],
Xinyun Cao [aut],
Shruthi Bandyadka [aut],
Anastasia Leshchyk [aut],
Tyler Faits [aut],
Mohammed Muzamil Khan [aut],
Zhe Wang [aut],
W. Evan Johnson [aut] (<https://orcid.org/0000-0002-6247-6595>),
Joshua David Campbell [aut]

Maintainer Yichen Wang <wangych@bu.edu>

R topics documented:

`.addSeuratToMetaDataSCE` *.addSeuratToMetaDataSCE Adds the input seurat object to the metadata slot of the input sce object (after removing the data matrices)*

Description

.addSeuratToMetaDataSCE Adds the input seurat object to the metadata slot of the input sce object (after removing the data matrices)

Usage

```
.addSeuratToMetaDataSCE(inSCE, seuratObject)
```

Arguments

`inSCE`	(sce) object to which seurat object should be added in the metadata slot (copy to)
`seuratObject`	seurat object which should be added to the metadata slot of sce object (copy from)

Value

Updated `SingleCellExperiment` object which now contains the seurat object in its metadata slot (excluding data matrices)

`.checkDiffExpResultExists` *Check if the specified MAST result in SingleCellExperiment object is complete. But does not garantee the biological correctness.*

Description

Check if the specified MAST result in SingleCellExperiment object is complete. But does not garantee the biological correctness.

Usage

```
.checkDiffExpResultExists(inSCE, useResult, labelBy = NULL)
```

Arguments

inSCE	SingleCellExperiment inherited object. a differential expression analysis function has to be run in advance.
useResult	character. A string specifying the analysisName used when running a differential expression analysis function.
labelBy	A single character for a column of rowData(inSCE) as where to search for the labeling text. Default NULL.

Value

Stop point if found

.computeSignificantPC *.computeSignificantPC Computes the significant principal components from an input sce object (must contain pca slot) using stdev*

Description

.computeSignificantPC Computes the significant principal components from an input sce object (must contain pca slot) using stdev

Usage

```
.computeSignificantPC(inSCE)
```

Arguments

inSCE	(sce) object with pca computed

Value

A numerical value indicating how many number of components are considered significant

---

.extractSCEAnnotation *Extract columns from row/colData and transfer to factors*

Description

Extract columns from row/colData and transfer to factors

Usage

```
.extractSCEAnnotation(inSCE, axis = NULL, columns = NULL, index = NULL)
```

Arguments

inSCE	SingleCellExperiment inherited object.
axis	Choose from "col" or "row".
columns	character vector. The columns needed to be extracted. If NULL, will return an empty data.frame with matched row names. Default NULL.
index	Valid index to subset the col/row.

Value

A data.frame object.

.formatDEAList *Helper function for differential expression analysis methods that accepts multiple ways of conditional subsetting and returns stable index format. Meanwhile it does all the input checkings.*

Description

Helper function for differential expression analysis methods that accepts multiple ways of conditional subsetting and returns stable index format. Meanwhile it does all the input checkings.

Usage

```
.formatDEAList(
  inSCE,
  useAssay,
  index1 = NULL,
  index2 = NULL,
  class = NULL,
  classGroup1 = NULL,
  classGroup2 = NULL,
  groupName1,
  groupName2,
  analysisName,
  covariates = NULL,
  overwrite = FALSE
)
```

Arguments

inSCE	SingleCellExperiment inherited object. Required.
useAssay	character. A string specifying which assay to use. Required.
index1	Any type of indices that can subset a SingleCellExperiment inherited object by cells. Specifies which cells are of interests. Default NULL.

index2	Any type of indices that can subset a SingleCellExperiment inherited object by cells. specifies the control group against those specified by index1. If NULL when using index specification, index1 cells will be compared with all other cells. Default NULL.
class	A vector/factor with ncol(inSCE) elements, or a character scalar that specifies a column name of colData(inSCE). Default NULL.
classGroup1	a vector specifying which "levels" given in class are of interests. Default NULL.
classGroup2	a vector specifying which "levels" given in class is the control group against those specified by classGroup1. If NULL when using annotation specification, classGroup1 cells will be compared with all other cells.
groupName1	A character scalar naming the group of interests. Required.
groupName2	A character scalar naming the control group. Required.
analysisName	A character scalar naming the DEG analysis. Required
covariates	A character vector of additional covariates used in linear regression methods such as Limma and DESeq2. Default NULL
overwrite	A logical scalar. Whether to overwrite result if exists. Default FALSE.

Value

A list object with part of formatted DE analysis information

Author(s)

Yichen Wang

.getComponentNames *.getComponentNames Creates a list of PC/IC components to populate the picker for PC/IC heatmap generation*

Description

.getComponentNames Creates a list of PC/IC components to populate the picker for PC/IC heatmap generation

Usage

```
.getComponentNames(maxComponents, component = c("PC", "IC"))
```

Arguments

maxComponents	Number of components to return for the picker
component	Which component to use. Choices are PC or IC.

Value

List of component names (appended with PC or IC)

.ggBar *Bar plot plotting tool.*

Description

Visualizes specified values via a violin plot.

Usage

```
.ggBar(
  y,
  groupBy = NULL,
  xlab = NULL,
  ylab = NULL,
  axisSize = 10,
  axisLabelSize = 10,
  dotSize = 0.5,
  transparency = 1,
  defaultTheme = TRUE,
  gridLine = FALSE,
  summary = NULL,
  title = NULL,
  titleSize = 15
)
```

Arguments

y	Numeric values to be plotted on y-axis.
groupBy	Groupings for each numeric value. A user may input a vector equal length to the number of the samples in the SingleCellExperiment object, or can be retrieved from the colData slot. Default NULL.
xlab	Character vector. Label for x-axis. Default NULL.
ylab	Character vector. Label for y-axis. Default NULL.
axisSize	Size of x/y-axis ticks. Default 10.
axisLabelSize	Size of x/y-axis labels. Default 10.
dotSize	Size of dots. Default 0.5.
transparency	Transparency of the dots, values will be 0-1. Default 1.
defaultTheme	Removes grid in plot and sets axis title size to 10 when TRUE. Default TRUE.
gridLine	Adds a horizontal grid line if TRUE. Will still be drawn even if defaultTheme is TRUE. Default FALSE.
summary	Adds a summary statistic, as well as a crossbar to the violin plot. Options are "mean" or "median". Default NULL.
title	Title of plot. Default NULL.
titleSize	Size of title of plot. Default 15.

Value

a ggplot of the reduced dimensions.

`.ggDensity` *Density plot plotting tool.*

Description

Visualizes values stored in the specified slot of a SingleCellExperiment object via a density plot.

Usage

```
.ggDensity(
  value,
  groupBy = NULL,
  xlab = NULL,
  ylab = NULL,
  baseSize = 12,
  axisSize = NULL,
  axisLabelSize = NULL,
  defaultTheme = TRUE,
  title = NULL,
  titleSize = NULL,
  combinePlot = "none",
  cutoff = NULL
)
```

Arguments

`value`	Numeric value that will be plotted via density plot.
`groupBy`	Groupings for each numeric value. A user may input a vector equal length to the number of the samples in the SingleCellExperiment object, or can be retrieved from the colData slot. Default NULL.
`xlab`	Character vector. Label for x-axis. Default NULL.
`ylab`	Character vector. Label for y-axis. Default NULL.
`baseSize`	The base font size for all text. Default 12. Can be overwritten by titleSize, axisSize, and axisLabelSize.
`axisSize`	Size of x/y-axis ticks. Default NULL.
`axisLabelSize`	Size of x/y-axis labels. Default NULL.
`defaultTheme`	Removes grid in plot and sets axis title size to 10 when TRUE. Default TRUE.
`title`	Title of plot. Default NULL.
`titleSize`	Size of title of plot. Default 15.

combinePlot	Must be either "all", "sample", or "none". "all" will combine all plots into a single .ggplot object, while "sample" will output a list of plots separated by sample. Default "none".
cutoff	Numeric value. The plot will be annotated with a vertical line if set. Default NULL.

Value

density plot, in .ggplot.

`.ggScatter` *Plot results of reduced dimensions data.*

Description

Plot results of reduced dimensions data and colors the plots by the input vector.

Usage

```
.ggScatter(
  inSCE,
  reducedDimName,
  sample = NULL,
  colorBy = NULL,
  groupBy = NULL,
  shape = NULL,
  conditionClass = NULL,
  labelClusters = FALSE,
  clusterLabelSize = 3.5,
  xlab = NULL,
  ylab = NULL,
  baseSize = 12,
  axisSize = NULL,
  axisLabelSize = NULL,
  dim1 = NULL,
  dim2 = NULL,
  bin = NULL,
  binLabel = NULL,
  dotSize = 0.5,
  transparency = 1,
  colorScale = NULL,
  colorLow = "white",
  colorMid = "gray",
  colorHigh = "blue",
  defaultTheme = TRUE,
  title = NULL,
  titleSize = NULL,
```

```
  legendTitle = NULL,
  legendTitleSize = NULL,
  legendSize = NULL,
  combinePlot = "none",
  plotLabels = NULL
)
```

Arguments

inSCE	Input SingleCellExperiment object with saved dimension reduction components or a variable with saved results. Required
reducedDimName	Saved dimension reduction name in the SingleCellExperiment object. Required.
sample	Character vector. Indicates which sample each cell belongs to.
colorBy	If provided, colors dots in the scatterplot based on value.
groupBy	If provided, facet wrap the scatterplot based on value.
shape	If provided, add shapes based on the value.
conditionClass	class of the annotation data used in colorBy. Options are NULL, "factor" or "numeric". If NULL, class will default to the original class. Default NULL.
labelClusters	Logical. Whether the cluster labels are plotted. Default FALSE.
clusterLabelSize	Numeric. Determines the size of cluster label when 'labelClusters' is set to TRUE. Default 3.5.
xlab	Character vector. Label for x-axis. Default NULL.
ylab	Character vector. Label for y-axis. Default NULL.
baseSize	The base font size for all text. Default 12. Can be overwritten by titleSize, axisSize, and axisLabelSize, legendSize, legendTitleSize.
axisSize	Size of x/y-axis ticks. Default NULL.
axisLabelSize	Size of x/y-axis labels. Default NULL.
dim1	1st dimension to be used for plotting. Can either be a string which specifies the name of the dimension to be plotted from reducedDims, or a numeric value which specifies the index of the dimension to be plotted. Default is NULL.
dim2	2nd dimension to be used for plotting. Can either be a string which specifies the name of the dimension to be plotted from reducedDims, or a numeric value which specifies the index of the dimension to be plotted. Default is NULL.
bin	Numeric vector. If single value, will divide the numeric values into the 'bin' groups. If more than one value, will bin numeric values using values as a cut point.
binLabel	Character vector. Labels for the bins created by the 'bin' parameter. Default NULL.
dotSize	Size of dots. Default 0.5.
transparency	Transparency of the dots, values will be 0-1. Default 1.
colorScale	Vector. Needs to be same length as the number of unique levels of 'colorBy'. Will be used only if conditionClass = "factor" or "character". Default NULL.

colorLow	Character. A color available from 'colors()'. The color will be used to signify the lowest values on the scale. Default 'white'. Will be used only if conditionClass = "numeric".
colorMid	Character. A color available from 'colors()'. The color will be used to signify the midpoint on the scale. Default 'gray'. Will be used only if conditionClass = "numeric".
colorHigh	Character. A color available from 'colors()'. The color will be used to signify the highest values on the scale. Default 'blue'. Will be used only if conditionClass = "numeric".
defaultTheme	Removes grid in plot and sets axis title size to 10 when TRUE. Default TRUE.
title	Title of plot. Default NULL.
titleSize	Size of title of plot. Default 15.
legendTitle	title of legend. Default NULL.
legendTitleSize	size of legend title. Default NULL.
legendSize	size of legend. Default NULL.
combinePlot	Must be either "all", "sample", or "none". "all" will combine all plots into a single .ggplot object, while "sample" will output a list of plots separated by sample. Default "none".
plotLabels	labels to each plot. If set to "default", will use the name of the samples as the labels. If set to "none", no label will be plotted.

Value

a ggplot of the reduced dimensions.

.ggViolin *Violin plot plotting tool.*

Description

Visualizes specified values via a violin plot.

Usage

```
.ggViolin(
  y,
  groupBy = NULL,
  violin = TRUE,
  boxplot = TRUE,
  dots = TRUE,
  xlab = NULL,
  ylab = NULL,
  baseSize = 12,
```

```
  axisSize = NULL,
  axisLabelSize = NULL,
  dotSize = 0.5,
  transparency = 1,
  defaultTheme = TRUE,
  gridLine = FALSE,
  summary = NULL,
  summaryTextSize = 3,
  combinePlot = "none",
  title = NULL,
  titleSize = NULL
)
```

Arguments

y	Numeric values to be plotted on y-axis.
groupBy	Groupings for each numeric value. A user may input a vector equal length to the number of the samples in the SingleCellExperiment object, or can be retrieved from the colData slot. Default NULL.
violin	Boolean. If TRUE, will plot the violin plot. Default TRUE.
boxplot	Boolean. If TRUE, will plot boxplots for each violin plot. Default TRUE.
dots	Boolean. If TRUE, will plot dots for each violin plot. Default TRUE.
xlab	Character vector. Label for x-axis. Default NULL.
ylab	Character vector. Label for y-axis. Default NULL.
baseSize	The base font size for all text. Default 12. Can be overwritten by titleSize, axisSize, and axisLabelSize.
axisSize	Size of x/y-axis ticks. Default NULL.
axisLabelSize	Size of x/y-axis labels. Default NULL.
dotSize	Size of dots. Default 0.5.
transparency	Transparency of the dots, values will be 0-1. Default 1.
defaultTheme	Removes grid in plot and sets axis title size to 10 when TRUE. Default TRUE.
gridLine	Adds a horizontal grid line if TRUE. Will still be drawn even if defaultTheme is TRUE. Default FALSE.
summary	Adds a summary statistic, as well as a crossbar to the violin plot. Options are "mean" or "median". Default NULL.
summaryTextSize	The text size of the summary statistic displayed above the violin plot. Default 3.
combinePlot	Must be either "all", "sample", or "none". "all" will combine all plots into a single .ggplot object, while "sample" will output a list of plots separated by sample. Default "none".
title	Title of plot. Default NULL.
titleSize	Size of title of plot. Default 15.

Value

a ggplot of the reduced dimensions.

`.sce2adata` *Coverts SingleCellExperiment object from R to anndata.AnnData object in Python*

Description

The AnnData object here can be saved to .h5ad file and read into Python interactive console. Mostly used senario is when you want to apply reticulated Python function, which only works with an anndata.AnnData object.

Usage

```
.sce2adata(SCE, useAssay = "counts")
```

Arguments

`SCE`	A SingleCellExperiment object.
`useAssay`	Character, default '"counts"'. The name of assay of interests that will be set as the primary matrix of the output AnnData. Available options can be listed by 'assayNames(SCE)'. Thee primary matrix will be saved in 'adata$X', Other assays will be stored in 'adata$obsm' together with the low-dimension representations (for now).

Value

A Python anndata.AnnData object

`.seuratGetVariableFeatures` *.seuratGetVariableFeatures Retrieves the requested number of variable feature names*

Description

.seuratGetVariableFeatures Retrieves the requested number of variable feature names

Usage

```
.seuratGetVariableFeatures(inSCE, numberOfFeatures)
```

Arguments

`inSCE`	(sce) object from which to extract the variable feature names
`numberOfFeatures`	numerical value indicating how many feature names should be retrieved (default is 100)

Value

list() of variable feature names

`.seuratInvalidate` *.seuratInvalidate Removes seurat data from the input SingleCellExperiment object specified by the task in the Seurat workflow.*

Description

.seuratInvalidate Removes seurat data from the input SingleCellExperiment object specified by the task in the Seurat workflow.

Usage

```
.seuratInvalidate(
  inSCE,
  scaleData = TRUE,
  varFeatures = TRUE,
  PCA = TRUE,
  ICA = TRUE,
  tSNE = TRUE,
  UMAP = TRUE,
  clusters = TRUE
)
```

Arguments

`inSCE`	Input `SingleCellExperiment` object to remove Seurat data from.
`scaleData`	Remove scaled data from seurat. Default `TRUE`.
`varFeatures`	Remove variable features from seurat. Default `TRUE`.
`PCA`	Remove PCA from seurat. Default `TRUE`.
`ICA`	Remove ICA from seurat. Default `TRUE`.
`tSNE`	Remove tSNE from seurat. Default `TRUE`.
`UMAP`	Remove UMAP from seurat. Default `TRUE`.
`clusters`	Remove clusters from seurat. Default `TRUE`.

Value

Updated SingleCellExperiment object containing the Seurat object in the metadata slot with the data removed

`.updateAssaySCE` *.updateAssaySCE Update/Modify/Add an assay in the provided SingleCellExperiment object from a Seurat object*

Description

.updateAssaySCE Update/Modify/Add an assay in the provided SingleCellExperiment object from a Seurat object

Usage

```
.updateAssaySCE(
  inSCE,
  seuratObject,
  assaySlotSCE,
  seuratDataSlot = "counts",
  seuratAssaySlot = "RNA"
)
```

Arguments

`inSCE`	Input SingleCellExperiment object
`seuratObject`	Input Seurat object
`assaySlotSCE`	Selected assay to update in the input SingleCellExperiment object
`seuratDataSlot`	Selected data slot from the Seurat object. Default `"counts"`.
`seuratAssaySlot`	Selected assay from Seurat object. Default `"RNA"`.

Value

A SingleCellExperiment object with data from Seurat object appended to the assay slot.

`calcEffectSizes` *Finds the effect sizes for all genes in the original dataset, regardless of significance.*

Description

Finds the effect sizes for all genes in the original dataset, regardless of significance.

Usage

```
calcEffectSizes(countMatrix, condition)
```

Arguments

countMatrix	Matrix. A simulated counts matrix, sans labels.
condition	Factor. The condition labels for the simulated cells. If more than 2 conditions are given, the first will be compared to all others by default.

Value

A vector of cohen's d effect sizes for each gene.

Examples

```
data("mouseBrainSubsetSCE")
res <- calcEffectSizes(assay(mouseBrainSubsetSCE, "counts"),
                       condition = colData(mouseBrainSubsetSCE)$level1class)
```

---

combineSCE *Combine a list of SingleCellExperiment objects as one SingleCellExperiment object*

Description

Combine a list of SingleCellExperiment objects as one SingleCellExperiment object

Usage

```
combineSCE(sceList, by.r = NULL, by.c = NULL, combined = TRUE)
```

Arguments

sceList	A list contains SingleCellExperiment objects. Currently, combineSCE function only support combining SCE objects with assay in dgCMatrix format. It does not support combining SCE with assay in delayedArray format.
by.r	Specifications of the columns used for merging rowData. If set as NULL, the rownames of rowData tables will be used to merging rowData. Default is NULL.
by.c	Specifications of the columns used for merging colData. If set as NULL, the rownames of colData tables will be used to merging colData. Default is NULL.
combined	logical; if TRUE, it will combine the list of SingleCellExperiment objects and return a SingleCellExperiment. If FALSE, it will return a list of SingleCellExperiment whose rowData, colData, assay and reducedDim data slot are compatible within SCE objects in the list. Default is TRUE.

Value

A SingleCellExperiment object which combines all objects in sceList. The colData is merged.

Examples

```
combinedsce <- combineSCE(list(sce,sce), by.r = NULL, by.c = NULL, combined = TRUE)
```

`computeHeatmap` *computeHeatmap The computeHeatmap method computes the heatmap visualization for a set of features against a set of dimensionality reduction components. This method uses the heatmap computation algorithm code from* `Seurat` *but plots the heatmap using* `ComplexHeatmap` *and* `cowplot` *libraries.*

Description

computeHeatmap The computeHeatmap method computes the heatmap visualization for a set of features against a set of dimensionality reduction components. This method uses the heatmap computation algorithm code from `Seurat` but plots the heatmap using `ComplexHeatmap` and `cowplot` libraries.

Usage

```
computeHeatmap(
  inSCE,
  useAssay,
  dims = 10,
  nfeatures = 30,
  cells = NULL,
  reduction = "pca",
  disp.min = -2.5,
  disp.max = 2.5,
  balanced = TRUE,
  nCol = NULL,
  externalReduction = NULL
)
```

Arguments

`inSCE`	Input `SingleCellExperiment` object.
`useAssay`	The assay to use for heatmap computation.
`dims`	Specify the number of dimensions to use for heatmap. Default `10`.
`nfeatures`	Specify the number of features to use for heatmap. Default is `30`.
`cells`	Specify the samples/cells to use for heatmap computation. Default is `NULL` which will utilize all samples in the assay.
`reduction`	Specify the reduction slot in the input object. Default is `"pca"`.
`disp.min`	Specify the minimum dispersion value to use for floor clipping of assay values. Default is `-2.5`.

disp.max	Specify the maximum dispersion value to use for ceiling clipping of assay values. Default is 2.5.
balanced	Specify if the number of of up-regulated and down-regulated features should be balanced. Default is TRUE.
nCol	Specify the number of columns in the output plot. Default is NULL which will auto-compute the number of columns.
externalReduction	Specify an external reduction if not present in the input object. This external reduction should be created using CreateDimReducObject function.

Value

Heatmap plot object.

`computeZScore` *Compute Z-Score*

Description

Computes Z-Score from an input count matrix using the formula ((x-mean(x))/sd(x)) for each gene across all cells. The input count matrix can either be a base matrix, dgCMatrix or a DelayedMatrix. Computations are performed using DelayedMatrixStats package to efficiently compute the Z-Score matrix.

Usage

```
computeZScore(counts)
```

Arguments

counts	matrix (base matrix, dgCMatrix or DelayedMatrix)

Value

z-score computed counts matrix (DelayedMatrix)

Examples

```
data(sce_chcl, package = "scds")
assay(sce_chcl, "countsZScore") <- computeZScore(assay(sce_chcl, "counts"))
```

`constructSCE` *Create SingleCellExperiment object from csv or txt input*

Description

Create SingleCellExperiment object from csv or txt input

Usage

```
constructSCE(data, samplename)
```

Arguments

`data`	A data.table object containing the count matrix.
`samplename`	The sample name of the data.

Value

A SingleCellExperiment object containing the count matrix.

`convertSCEToSeurat` *convertSCEToSeurat Converts sce object to seurat while retaining all assays and metadata*

Description

convertSCEToSeurat Converts sce object to seurat while retaining all assays and metadata

Usage

```
convertSCEToSeurat(
  inSCE,
  countsAssay = NULL,
  normAssay = NULL,
  scaledAssay = NULL,
  copyColData = FALSE,
  copyReducedDim = FALSE,
  copyDecontX = FALSE,
  pcaReducedDim = NULL,
  icaReducedDim = NULL,
  tsneReducedDim = NULL,
  umapReducedDim = NULL
)
```

Arguments

inSCE	A SingleCellExperiment object to convert to a Seurat object.
countsAssay	Which assay to use from sce object for raw counts. Default NULL.
normAssay	Which assay to use from sce object for normalized data. Default NULL.
scaledAssay	Which assay to use from sce object for scaled data. Default NULL.
copyColData	Boolean. Whether copy 'colData' of SCE object to the 'meta.data' of Seurat object. Default FALSE.
copyReducedDim	Boolean. Whether copy 'reducedDims' of the SCE object to the 'reductions' of Seurat object. Default FALSE.
copyDecontX	Boolean. Whether copy 'decontXcounts' assay of the SCE object to the 'assays' of Seurat object. Default TRUE.
pcaReducedDim	Specify a character value indicating the name of the reducedDim to store as default pca computation in the output seurat object. Default is NULL which will not store any reducedDim as the default pca. This will only work when copyReducedDim parameter is set to TRUE.
icaReducedDim	Specify a character value indicating the name of the reducedDim to store as default ica computation in the output seurat object. Default is NULL which will not store any reducedDim as the default ica. This will only work when copyReducedDim parameter is set to TRUE.
tsneReducedDim	Specify a character value indicating the name of the reducedDim to store as default tsne computation in the output seurat object. Default is NULL which will not store any reducedDim as the default tsne. This will only work when copyReducedDim parameter is set to TRUE.
umapReducedDim	Specify a character value indicating the name of the reducedDim to store as default umap computation in the output seurat object. Default is NULL which will not store any reducedDim as the default umap. This will only work when copyReducedDim parameter is set to TRUE.

Value

Updated seurat object that contains all data from the input sce object

Examples

```
data(scExample, package = "singleCellTK")
seurat <- convertSCEToSeurat(sce)
```

---

convertSeuratToSCE *convertSeuratToSCE Converts the input seurat object to a sce object*

Description

convertSeuratToSCE Converts the input seurat object to a sce object

Usage

```
convertSeuratToSCE(
  seuratObject,
  normAssayName = "seuratNormData",
  scaledAssayName = "seuratScaledData"
)
```

Arguments

seuratObject	Input Seurat object
normAssayName	Name of assay to store the normalized data. Default "seuratNormData".
scaledAssayName	Name of assay to store the scaled data. Default "seuratScaledData".

Value

SingleCellExperiment output object

Examples

```
data(scExample, package = "singleCellTK")
seurat <- convertSCEToSeurat(sce)
sce <- convertSeuratToSCE(seurat)
```

---

dataAnnotationColor *Generate distinct colors for all categorical col/rowData entries. Character columns will be considered as well as all-integer columns. Any column with all-distinct values will be excluded.*

Description

Generate distinct colors for all categorical col/rowData entries. Character columns will be considered as well as all-integer columns. Any column with all-distinct values will be excluded.

Usage

```
dataAnnotationColor(inSCE, axis = NULL, colorGen = distinctColors)
```

Arguments

inSCE	SingleCellExperiment inherited object.
axis	Choose from "col" or "row".
colorGen	A function that generates color code vector by giving an integer for the number of colors. Alternatively, rainbow. Default distinctColors.

Value

A `list` object containing distinct colors mapped to all possible categorical entries in `rowData(inSCE)` or `colData(inSCE)`.

Author(s)

Yichen Wang

`dedupRowNames` *Deduplicate the rownames of a matrix or SingleCellExperiment object Adds '-1', '-2', ... '-i' to multiple duplicated rownames, and in place replace the unique rownames, store unique rownames in* `rowData`*, or return the unique rownames as character vecetor.*

Description

Deduplicate the rownames of a matrix or SingleCellExperiment object Adds '-1', '-2', ... '-i' to multiple duplicated rownames, and in place replace the unique rownames, store unique rownames in `rowData`, or return the unique rownames as character vecetor.

Usage

```
dedupRowNames(x, as.rowData = FALSE, return.list = FALSE)
```

Arguments

`x`	A matrix like or /linkS4classSingleCellExperiment object, on which we can apply `rownames()` to and has duplicated rownames.
`as.rowData`	Only applicable when `x` is a /linkS4classSingleCellExperiment object. When set to `TRUE`, will insert a new column called `"rownames.uniq"` to `rowData(x)`, with the deduplicated rownames.
`return.list`	When set to `TRUE`, will return a character vector with deduplicated rownames.

Value

By default, a matrix or /linkS4classSingleCellExperiment object with rownames deduplicated. When `x` is a /linkS4classSingleCellExperiment and `as.rowData` is set to `TRUE`, will return `x` with `rowData` updated. When `return.list` is set to `TRUE`, will return a character vector with the deduplicated rownames.

Examples

```
data("scExample", package = "singleCellTK")
sce <- dedupRowNames(sce)
```

`detectCellOutlier` *Detecting outliers within the SingleCellExperiment object.*

Description

A wrapper function for isOutlier. Identify outliers from numeric vectors stored in the SingleCellExperiment object.

Usage

```
detectCellOutlier(
  inSCE,
  slotName,
  itemName,
  sample = NULL,
  nmads = 3,
  type = "both",
  overwrite = TRUE
)
```

Arguments

`inSCE`	A SingleCellExperiment object.
`slotName`	Desired slot of SingleCellExperiment used for plotting. Possible options: "assays", "colData", "metadata", "reducedDims". Required.
`itemName`	Desired vector within the slot used for plotting. Required.
`sample`	A single character specifying a name that can be found in `colData(inSCE)` to directly use the cell annotation; or a character vector with as many elements as cells to indicates which sample each cell belongs to. Default NULL. decontX will be run on cells from each sample separately.
`nmads`	Integer. Number of median absolute deviation. Parameter may be adjusted for more lenient or stringent outlier cutoff. Default 3.
`type`	Character. Type/direction of outlier detection; whether the lower/higher outliers should be detected, or both. Options are "both", "lower", "higher".
`overwrite`	Boolean. If TRUE, and this function has previously generated an outlier decision on the same itemName, the outlier decision will be overwritten. Default TRUE.

Value

A SingleCellExperiment object with ” added to the colData slot. Additionally, the decontaminated counts will be added as an assay called 'decontXCounts'.

Examples

```
data(scExample, package = "singleCellTK")
sce <- subsetSCECols(sce, colData = "type != 'EmptyDroplet'")
sce <- runDecontX(sce[,sample(ncol(sce),20)])
sce <- detectCellOutlier(sce, slotName = "colData", sample = sce$sample,
 nmads = 4, itemName = "decontX_contamination", type = "both")
```

`diffAbundanceFET` *Calculate Differential Abundance with FET*

Description

Calculate Differential Abundance with FET

Usage

```
diffAbundanceFET(inSCE, cluster, variable, control, case, analysisName)
```

Arguments

`inSCE`	A `SingleCellExperiment` object.
`cluster`	A single `character`, specifying the name to store the cluster label in `colData`.
`variable`	A single `character`, specifying the name to store the phenotype labels in `colData`.
`control`	`character`. Specifying one or more categories that can be found in the vector specified by `variable`.
`case`	`character`. Specifying one or more categories that can be found in the vector specified by `variable`.
`analysisName`	A single `character`. Will be used for naming the result table, which will be saved in metadata slot.

Details

This function will calculate the cell counting and fraction by dividing all cells to groups specified by the arguments, together with statistical summary by performing Fisher Exact Tests (FET).

Value

The original `SingleCellExperiment` object with `metadata(inSCE)` updated with a list `diffAbundanceFET`, containing a new `data.frame` for the analysis result, named by `analysisName`. The `data.frame` contains columns for number and fraction of cells that belong to different cases, as well as "Odds_Ratio", "PValue" and "FDR".

Examples

```
data("mouseBrainSubsetSCE", package = "singleCellTK")
mouseBrainSubsetSCE <- diffAbundanceFET(inSCE = mouseBrainSubsetSCE,
                                                cluster = "tissue",
                                                variable = "level1class",
                                                case = "oligodendrocytes",
                                                control = "microglia",
                                                analysisName = "diffAbundFET")
```

`discreteColorPalette` *Generate given number of color codes*

Description

Three different generation methods are wrapped, including `distinctColors`, [randomcoloR](SCTK_PerformingQC_Cell_V
and the `ggplot` default color generation.

Usage

```
discreteColorPalette(
  n,
  palette = c("random", "ggplot", "celda"),
  seed = 12345,
  ...
)
```

Arguments

`n`	An integer, the number of color codes to generate.
`palette`	A single character string. Select the method, available options are `"ggplot"`, `"celda"` and `"random"`. Default `"random"`.
`seed`	An integer. Set the seed for random process that happens only in "random" generation. Default `12345`.
`...`	Other arguments that are passed to the internal function, according to the method selected.

Value

A character vector of `n` hex color codes.

Examples

```
discreteColorPalette(n = 3)
```

distinctColors *Generate a distinct palette for coloring different clusters*

Description

Generate a distinct palette for coloring different clusters

Usage

```
distinctColors(
  n,
 hues = c("red", "cyan", "orange", "blue", "yellow", "purple", "green", "magenta"),
  saturation.range = c(0.7, 1),
  value.range = c(0.7, 1)
)
```

Arguments

n	Integer; Number of colors to generate
hues	Character vector of R colors available from the colors() function. These will be used as the base colors for the clustering scheme. Different saturations and values (i.e. darkness) will be generated for each hue.
saturation.range	Numeric vector of length 2 with values between 0 and 1. Default: c(0.25, 1)
value.range	Numeric vector of length 2 with values between 0 and 1. Default: c(0.5, 1)

Value

A vector of distinct colors that have been converted to HEX from HSV.

Examples

```
distinctColors(10)
```

downSampleCells *Estimate numbers of detected genes, significantly differentially expressed genes, and median significant effect size*

Description

Estimate numbers of detected genes, significantly differentially expressed genes, and median significant effect size

Usage

```
downSampleCells(
  originalData,
  useAssay = "counts",
  minCountDetec = 10,
  minCellsDetec = 3,
  minCellnum = 10,
  maxCellnum = 1000,
  realLabels,
  depthResolution = 10,
  iterations = 10,
  totalReads = 1e+06
)
```

Arguments

originalData	The SingleCellExperiment object storing all assay data from the shiny app.
useAssay	Character. The name of the assay to be used for subsampling.
minCountDetec	Numeric. The minimum number of reads found for a gene to be considered detected.
minCellsDetec	Numeric. The minimum number of cells a gene must have at least 1 read in for it to be considered detected.
minCellnum	Numeric. The minimum number of virtual cells to include in the smallest simulated dataset.
maxCellnum	Numeric. The maximum number of virtual cells to include in the largest simulated dataset
realLabels	Character. The name of the condition of interest. Must match a name from sample data. If only two factors present in the corresponding colData, will default to t-test. If multiple factors, will default to ANOVA.
depthResolution	Numeric. How many different read depth should the script simulate? Will simulate a number of experimental designs ranging from 10 reads to maxReadDepth, with logarithmic spacing.
iterations	Numeric. How many times should each experimental design be simulated?
totalReads	Numeric. How many aligned reads to put in each simulated dataset.

Value

A 3-dimensional array, with dimensions = c(iterations, depthResolution, 3). [,,1] contains the number of detected genes in each simulated dataset, [,,2] contains the number of significantly differentially expressed genes in each simulation, and [,,3] contains the mediansignificant effect size in each simulation. If no genes are significantly differentially expressed, the median effect size defaults to infinity.

Examples

```
data("mouseBrainSubsetSCE")
subset <- mouseBrainSubsetSCE[seq(100),]
res <- downSampleCells(subset,
                       realLabels = "level1class",
                       iterations=2)
```

`downSampleDepth` *Estimate numbers of detected genes, significantly differentially expressed genes, and median significant effect size*

Description

Estimate numbers of detected genes, significantly differentially expressed genes, and median significant effect size

Usage

```
downSampleDepth(
  originalData,
  useAssay = "counts",
  minCount = 10,
  minCells = 3,
  maxDepth = 1e+07,
  realLabels,
  depthResolution = 10,
  iterations = 10
)
```

Arguments

`originalData`	SingleCellExperiment object storing all assay data from the shiny app.
`useAssay`	Character. The name of the assay to be used for subsampling.
`minCount`	Numeric. The minimum number of reads found for a gene to be considered detected.
`minCells`	Numeric. The minimum number of cells a gene must have at least 1 read in for it to be considered detected.
`maxDepth`	Numeric. The highest number of total reads to be simulated.
`realLabels`	Character. The name of the condition of interest. Must match a name from sample data.
`depthResolution`	Numeric. How many different read depth should the script simulate? Will simulate a number of experimental designs ranging from 10 reads to maxReadDepth, with logarithmic spacing.
`iterations`	Numeric. How many times should each experimental design be simulated?

Value

A 3-dimensional array, with dimensions = c(iterations, depthResolution, 3). [,,1] contains the number of detected genes in each simulated dataset, [,,2] contains the number of significantly differentially expressed genes in each simulation, and [,,3] contains the mediansignificant effect size in each simulation. If no genes are significantly differentially expressed, the median effect size defaults to infinity.

Examples

```
data("mouseBrainSubsetSCE")
subset <- mouseBrainSubsetSCE[seq(1000),]
res <- downSampleDepth(subset,
                       realLabels = "level1class",
                       iterations=2)
```

`enrichRSCE` *enrichR Given a list of genes this function runs the enrichR() to perform Gene enrichment*

Description

enrichR Given a list of genes this function runs the enrichR() to perform Gene enrichment

Usage

```
enrichRSCE(inSCE, glist, db = NULL)
```

Arguments

`inSCE`	Input SingleCellExperiment object.
`glist`	selected genes for enrichment analysis using enrichR(). Required
`db`	selected database name from the enrichR database list. if NULL then enrichR will be run on all the available databases on the enrichR database.

Value

enrichRSCE(): returns a data.frame of enrichment terms overlapping in the respective databases along with p-values, z-scores etc.,

Examples

```
enrichRSCE(mouseBrainSubsetSCE, "Cmtm5", "GO_Cellular_Component_2017")
```

expData *expData Get data item from an input* SingleCellExperiment *object. The data item can be an* assay*,* altExp *(subset) or a* reducedDim*, which is retrieved based on the name of the data item.*

Description

expData Get data item from an input SingleCellExperiment object. The data item can be an assay, altExp (subset) or a reducedDim, which is retrieved based on the name of the data item.

Usage

```
expData(inSCE, assayName)
```

Arguments

inSCE	Input SingleCellExperiment object.
assayName	Specify the name of the data item to retrieve.

Value

Specified data item.

expData,ANY,character-method *expData Get data item from an input* SingleCellExperiment *object. The data item can be an* assay*,* altExp *(subset) or a* reducedDim*, which is retrieved based on the name of the data item.*

Description

expData Get data item from an input SingleCellExperiment object. The data item can be an assay, altExp (subset) or a reducedDim, which is retrieved based on the name of the data item.

Usage

```
## S4 method for signature 'ANY,character'
expData(inSCE, assayName)
```

Arguments

inSCE	Input SingleCellExperiment object.
assayName	Specify the name of the data item to retrieve.

Value

Specified data item.

expData<- *expData Store data items using tags to identify the type of data item stored. To be used as a replacement for assay<- setter function but with additional parameter to set a tag to a data item.*

Description

expData Store data items using tags to identify the type of data item stored. To be used as a replacement for assay<- setter function but with additional parameter to set a tag to a data item.

Usage

```
expData(inSCE, assayName, tag = NULL, altExp = FALSE) <- value
```

Arguments

inSCE	Input SingleCellExperiment object.
assayName	Specify the name of the input assay.
tag	Specify the tag to store against the input assay. Default is NULL, which will set the tag to "uncategorized".
altExp	A logical value indicating if the input assay is a altExp or a subset assay.
value	An input matrix-like value to store in the SCE object.

Value

A SingleCellExperiment object containing the newly stored data.

expData<-,ANY,character,CharacterOrNullOrMissing,logical-method *expData Store data items using tags to identify the type of data item stored. To be used as a replacement for assay<- setter function but with additional parameter to set a tag to a data item.*

Description

expData Store data items using tags to identify the type of data item stored. To be used as a replacement for assay<- setter function but with additional parameter to set a tag to a data item.

Usage

```
## S4 replacement method for signature 'ANY,character,CharacterOrNullOrMissing,logical'
expData(inSCE, assayName, tag = NULL, altExp = FALSE) <- value
```

Arguments

inSCE	Input SingleCellExperiment object.
assayName	Specify the name of the input assay.
tag	Specify the tag to store against the input assay. Default is NULL, which will set the tag to "uncategorized".
altExp	A logical value indicating if the input assay is a altExp or a subset assay.
value	An input matrix-like value to store in the SCE object.

Value

A SingleCellExperiment object containing the newly stored data.

---

expDataNames *expDataNames Get names of all the data items in the input* SingleCellExperiment *object including assays, altExps and reducedDims.*

Description

expDataNames Get names of all the data items in the input SingleCellExperiment object including assays, altExps and reducedDims.

Usage

```
expDataNames(inSCE)
```

Arguments

inSCE	Input SingleCellExperiment object.

Value

A combined vector of assayNames, altExpNames and reducedDimNames.

expDataNames,ANY-method *expDataNames Get names of all the data items in the input* SingleCellExperiment *object including assays, altExps and reducedDims.*

Description

expDataNames Get names of all the data items in the input SingleCellExperiment object including assays, altExps and reducedDims.

Usage

```
## S4 method for signature 'ANY'
expDataNames(inSCE)
```

Arguments

inSCE	Input SingleCellExperiment object.

Value

A combined vector of assayNames, altExpNames and reducedDimNames.

expDeleteDataTag *expDeleteDataTag Remove tag against an input data from the stored tag information in the metadata of the input object.*

Description

expDeleteDataTag Remove tag against an input data from the stored tag information in the metadata of the input object.

Usage

```
expDeleteDataTag(inSCE, assay)
```

Arguments

inSCE	Input SingleCellExperiment object.
assay	Name of the assay or the data item against which a tag should be removed.

Value

The input SingleCellExperiment object with tag information removed from the metadata slot.

exportSCE *Export data in SingleCellExperiment object*

Description

Export data in SingleCellExperiment object

Usage

```
exportSCE(
  inSCE,
  samplename = "sample",
  directory = "./",
  type = "Cells",
  format = c("SCE", "AnnData", "FlatFile", "HTAN", "Seurat")
)
```

Arguments

inSCE	A SingleCellExperiment object that contains the data. QC metrics are stored in colData of the singleCellExperiment object.
samplename	Sample name. This will be used as name of subdirectories and the prefix of flat file output. Default is 'sample'.
directory	Output directory. Default is './'.
type	Type of data. The type of data stored in SingleCellExperiment object. It can be 'Droplets'(raw droplets matrix) or 'Cells' (cells matrix).
format	The format of output. It currently supports flat files, rds files and python h5 files. It can output multiple formats. Default: c("SCE", "AnnData", "FlatFile", "HTAN").

Value

Generates a file containing data from inSCE, in specified format.

Examples

```
data(scExample)
## Not run:
exportSCE(sce, format = "SCE")

## End(Not run)
```

`exportSCEtoAnnData` *Export a SingleCellExperiment R object as Python annData object*

Description

Writes all assays, colData, rowData, reducedDims, and altExps objects in a SingleCellExperiment to a Python annData object in the .h5ad format All parameters of Anndata.write_h5ad function (https://icb-anndata.readthedocs-hosted.com/en/stable/anndata.AnnData.write_h5ad.html) are available as parameters to this export function and set to defaults. Defaults can be overridden at function call.

Usage

```
exportSCEtoAnnData(
  sce,
  useAssay = "counts",
  outputDir = "./",
  prefix = "sample",
  overwrite = TRUE,
  compression = c("gzip", "lzf", "None"),
  compressionOpts = NULL,
  forceDense = FALSE
)
```

Arguments

`sce`	SingleCellExperiment R object to be exported.
`useAssay`	Character. The name of assay of interests that will be set as the primary matrix of the output AnnData. Default `"counts"`.
`outputDir`	Path to the directory where .h5ad outputs will be written. Default is the current working directory.
`prefix`	Prefix to use for the name of the output file. Default `"sample"`.
`overwrite`	Boolean. Default `TRUE`.
`compression`	If output file compression is required, this variable accepts 'gzip', 'lzf' or "None" as inputs. Default `"gzip"`.
`compressionOpts`	Integer. Sets the compression level
`forceDense`	Default `False` Write sparse data as a dense matrix. Refer `anndata.write_h5ad` documentation for details. Default `NULL`.

Value

Generates a Python anndata object containing data from `inSCE`.

Examples

```
data(sce_chcl, package = "scds")
## Not run:
exportSCEtoAnnData(sce=sce_chcl, compression="gzip")

## End(Not run)
```

`exportSCEtoFlatFile` *Export a SingleCellExperiment object to flat text files*

Description

Writes all assays, colData, rowData, reducedDims, and altExps objects in a SingleCellExperiment to text files. The items in the 'metadata' slot remain stored in list and are saved in an RDS file.

Usage

```
exportSCEtoFlatFile(
  sce,
  outputDir = "./",
  overwrite = TRUE,
  gzipped = TRUE,
  prefix = "SCE"
)
```

Arguments

`sce`	SingleCellExperiment object to be exported.
`outputDir`	Name of the directory to store the exported file(s).
`overwrite`	Boolean. Whether to overwrite the output files. Default `TRUE`.
`gzipped`	Boolean. `TRUE` if the output files are to be gzip compressed. `FALSE` otherwise. Default `TRUE`.
`prefix`	Prefix of file names.

Value

Generates text files containing data from `inSCE`.

Examples

```
data(sce_chcl, package = "scds")
## Not run:
exportSCEtoFlatFile(sce_chcl, "sce_chcl")

## End(Not run)
```

`exportSCEToSeurat` *Export data in Seurat object*

Description

Export data in Seurat object

Usage

```
exportSCEToSeurat(
  inSCE,
  prefix = "sample",
  outputDir = "./",
  overwrite = TRUE,
  copyColData = TRUE,
  copyReducedDim = TRUE,
  copyDecontX = TRUE
)
```

Arguments

`inSCE`	A SingleCellExperiment object that contains the data. QC metrics are stored in colData of the singleCellExperiment object.
`prefix`	Prefix to use for the name of the output file. Default `"sample"`.
`outputDir`	Path to the directory where outputs will be written. Default is the current working directory.
`overwrite`	Boolean. Whether overwrite the output if it already exists in the outputDir. Default `TRUE`.
`copyColData`	Boolean. Whether copy 'colData' of SCE object to the 'meta.data' of Seurat object. Default `TRUE`.
`copyReducedDim`	Boolean. Whether copy 'reducedDims' of the SCE object to the 'reductions' of Seurat object. Default `TRUE`.
`copyDecontX`	Boolean. Whether copy 'decontXcounts' assay of the SCE object to the 'assays' of Seurat object. Default `TRUE`.

Value

Generates a Seurat object containing data from `inSCE`.

`expSetDataTag` *expSetDataTag Set tag to an assay or a data item in the input SCE object.*

Description

expSetDataTag Set tag to an assay or a data item in the input SCE object.

Usage

```
expSetDataTag(inSCE, assayType, assays)
```

Arguments

`inSCE`	Input `SingleCellExperiment` object.
`assayType`	Specify a `character(1)` value as a tag that should be set against a data item.
`assays`	Specify name(s) `character()` of data item(s) against which the tag should be set.

Value

The input `SingleCellExperiment` object with tag information stored in the metadata slot.

`expTaggedData` *expTaggedData Returns a list of names of data items from the input* `SingleCellExperiment` *object based upon the input parameters.*

Description

expTaggedData Returns a list of names of data items from the input `SingleCellExperiment` object based upon the input parameters.

Usage

```
expTaggedData(
  inSCE,
  tags = NULL,
  redDims = FALSE,
  recommended = NULL,
  showTags = TRUE
)
```

Arguments

inSCE	Input SingleCellExperiment object.
tags	A character() value indicating if the data items should be returned separated by the specified tags. Default is NULL indicating that returned names of the data items are simply returned as a list with default tag as "uncategorized".
redDims	A logical value indicating if reducedDims should be returned as well separated with 'redDims' tag.
recommended	A character() vector indicating the tags that should be displayed as recommended. Default is NULL.
showTags	A logical value indicating if the tags should be shown. If FALSE, output is just a simple list, not separated by tags.

Value

A list of names of data items specified by the other parameters.

---

featureIndex *Retrieve row index for a set of features*

Description

This will return indices of features among the rownames or rowData of a data.frame, matrix, or a SummarizedExperiment object including a SingleCellExperiment. Partial matching (i.e. grepping) can be used by setting exactMatch = FALSE.

Usage

```
featureIndex(
  features,
  inSCE,
  by = "rownames",
  exactMatch = TRUE,
  removeNA = FALSE,
  errorOnNoMatch = TRUE,
  warningOnPartialMatch = TRUE
)
```

Arguments

features	Character vector of feature names to find in the rows of inSCE.
inSCE	A data.frame, matrix, or SingleCellExperiment object to search.
by	Character. Where to search for features in inSCE. If set to "rownames" then the features will be searched for among rownames(inSCE). If inSCE inherits from class SummarizedExperiment, then by can be one of the fields in the row annotation data.frame (i.e. one of colnames(rowData(inSCE))).

exactMatch Boolean. Whether to only identify exact matches or to identify partial matches using grep.

removeNA Boolean. If set to FALSE, features not found in inSCE will be given NA and the returned vector will be the same length as features. If set to TRUE, then the NA values will be removed from the returned vector. Default FALSE.

errorOnNoMatch Boolean. If TRUE, an error will be given if no matches are found. If FALSE, an empty vector will be returned if removeNA is set to TRUE or a vector of NA if removeNA is set to FALSE. Default TRUE.

warningOnPartialMatch
Boolean. If TRUE, a warning will be given if some of the entries in features were not found in inSCE. The warning will list the features not found. Default TRUE.

Value

A vector of row indices for the matching features in inSCE.

Author(s)

Yusuke Koga, Joshua D. Campbell

See Also

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

Examples

```
data(scExample)
ix <- featureIndex(features = c("MT-CYB", "MT-ND2"),
                           inSCE = sce,
                           by = "feature_name")
```

---

findMarkerDiffExp *Find the marker gene set for each cluster With an input SingleCellExperiment object and specifying the clustering labels, this function iteratively call the differential expression analysis on each cluster against all the others.*

Description

Find the marker gene set for each cluster With an input SingleCellExperiment object and specifying the clustering labels, this function iteratively call the differential expression analysis on each cluster against all the others.

Usage

```
findMarkerDiffExp(
  inSCE,
  useAssay = "logcounts",
  method = c("wilcox", "MAST", "DESeq2", "Limma", "ANOVA"),
  cluster = "cluster",
  covariates = NULL,
  log2fcThreshold = 0.25,
  fdrThreshold = 0.05,
  minClustExprPerc = 0.6,
  maxCtrlExprPerc = 0.4,
  minMeanExpr = 0.5
)
```

Arguments

inSCE	SingleCellExperiment inherited object.
useAssay	character. A string specifying which assay to use for the MAST calculations. Default "logcounts".
method	A single character for specific differential expression analysis method. Choose from 'wilcox', 'MAST', 'DESeq2', 'Limma', and 'ANOVA'. Default "wilcox".
cluster	One single character to specify a column in colData(inSCE) for the clustering label. Alternatively, a vector or a factor is also acceptable. Default "cluster".
covariates	A character vector of additional covariates to use when building the model. All covariates must exist in names(colData(inSCE)). Not applicable when method is "MAST" method. Default NULL.
log2fcThreshold	Only out put DEGs with the absolute values of log2FC larger than this value. Default NULL
fdrThreshold	Only out put DEGs with FDR value smaller than this value. Default 1
minClustExprPerc	A numeric scalar. The minimum cutoff of the percentage of cells in the cluster of interests that expressed the marker gene. Default 0.7.
maxCtrlExprPerc	A numeric scalar. The maximum cutoff of the percentage of cells out of the cluster (control group) that expressed the marker gene. Default 0.4.
minMeanExpr	A numeric scalar. The minimum cutoff of the mean expression value of the marker in the cluster of interests. Default 1.

Value

The input SingleCellExperiment object with metadata(inSCE)$findMarker updated with a data.table of the up- regulated DEGs for each cluster.

Examples

```
data("mouseBrainSubsetSCE", package = "singleCellTK")
mouseBrainSubsetSCE <- findMarkerDiffExp(mouseBrainSubsetSCE,
                                         useAssay = "logcounts",
                                         cluster = "level1class")
```

`findMarkerTopTable` *Fetch the table of top markers that pass the filtering*

Description

Fetch the table of top markers that pass the filtering

Usage

```
findMarkerTopTable(
  inSCE,
  log2fcThreshold = 1,
  fdrThreshold = 0.05,
  minClustExprPerc = 0.7,
  maxCtrlExprPerc = 0.4,
  minMeanExpr = 1,
  topN = 10
)
```

Arguments

`inSCE`	SingleCellExperiment inherited object.
`log2fcThreshold`	Only use DEGs with the absolute values of log2FC larger than this value. Default `1`
`fdrThreshold`	Only use DEGs with FDR value smaller than this value. Default `0.05`
`minClustExprPerc`	A numeric scalar. The minimum cutoff of the percentage of cells in the cluster of interests that expressed the marker gene. Default `0.7`.
`maxCtrlExprPerc`	A numeric scalar. The maximum cutoff of the percentage of cells out of the cluster (control group) that expressed the marker gene. Default `0.4`.
`minMeanExpr`	A numeric scalar. The minimum cutoff of the mean expression value of the marker in the cluster of interests. Default `1`.
`topN`	An integer. Only to fetch this number of top markers for each cluster in maximum, in terms of log2FC value. Use `NULL` to cancel the top N subscription. Default `10`.

Details

Users have to run `findMarkerDiffExp()` prior to using this function to extract a top marker table.

Value

An organized `data.frame` object, with the top marker gene information.

Examples

```
data("mouseBrainSubsetSCE", package = "singleCellTK")
mouseBrainSubsetSCE <- findMarkerDiffExp(mouseBrainSubsetSCE,
                                         useAssay = "logcounts",
                                         cluster = "level1class")
findMarkerTopTable(mouseBrainSubsetSCE)
```

---

generateHTANMeta *Generate HTAN manifest file for droplet and cell count data*

Description

Generate HTAN manifest file for droplet and cell count data

Usage

```
generateHTANMeta(
  dropletSCE = NULL,
  cellSCE = NULL,
  samplename,
  htan_patient_id,
  dir,
  dataType = c("Droplet", "Cell", "Both")
)
```

Arguments

`dropletSCE`	A SingleCellExperiment object containing droplet count matrix data
`cellSCE`	A SingleCellExperiment object containing cell count matrix data
`samplename`	The sample name of the SingleCellExperiment objects
`htan_patient_id`	The HTAN patient id of the sample in SingleCellExperiment object
`dir`	The output directory of the SCTK QC pipeline.
`dataType`	Type of the input data. It can be one of "Droplet", "Cell" or "Both".

Value

A SingleCellExperiment object which combines all objects in sceList. The colData is merged.

generateMeta *Generate HTAN manifest file for droplet and cell count data*

Description

Generate HTAN manifest file for droplet and cell count data

Usage

```
generateMeta(
  dropletSCE = NULL,
  cellSCE = NULL,
  samplename,
  dir,
  HTAN = TRUE,
  dataType = c("Droplet", "Cell", "Both")
)
```

Arguments

dropletSCE	A SingleCellExperiment object containing droplet count matrix data
cellSCE	A SingleCellExperiment object containing cell count matrix data
samplename	The sample name of the SingleCellExperiment objects
dir	The output directory of the SCTK QC pipeline.
HTAN	Whether generates manifest file including HTAN specific ID (HTAN Biospecimen ID, HTAN parent file ID and HTAN patient ID). Default is TRUE.
dataType	Type of the input data. It can be one of "Droplet", "Cell" or "Both".

Value

A SingleCellExperiment object which combines all objects in sceList. The colData is merged.

generateSimulatedData *Generates a single simulated dataset, bootstrapping from the input counts matrix.*

Description

Generates a single simulated dataset, bootstrapping from the input counts matrix.

Usage

```
generateSimulatedData(totalReads, cells, originalData, realLabels)
```

Arguments

totalReads	Numeric. The total number of reads in the simulated dataset, to be split between all simulated cells.
cells	Numeric. The number of virtual cells to simulate.
originalData	Matrix. The original raw read count matrix. When used within the Shiny app, this will be assay(SCEsetObject, "counts").
realLabels	Factor. The condition labels for differential expression. If only two factors present, will default to t-test. If multiple factors, will default to ANOVA.

Value

A simulated counts matrix, the first row of which contains the 'true' labels for each virtual cell.

Examples

```
data("mouseBrainSubsetSCE")
res <- generateSimulatedData(
         totalReads = 1000, cells=10,
         originalData = assay(mouseBrainSubsetSCE, "counts"),
         realLabels = colData(mouseBrainSubsetSCE)[, "level1class"])
```

---

getBiomarker *Given a list of genes and a SingleCellExperiment object, return the binary or continuous expression of the genes.*

Description

Given a list of genes and a SingleCellExperiment object, return the binary or continuous expression of the genes.

Usage

```
getBiomarker(
  inSCE,
  gene,
  binary = "Binary",
  useAssay = "counts",
  featureLocation = NULL,
  featureDisplay = NULL
)
```

Arguments

inSCE	Input SingleCellExperiment object.
gene	gene list
binary	"Binary" for binary expression or "Continuous" for a gradient. Default: "Binary"
useAssay	Indicates which assay to use. The default is "counts".
featureLocation	Indicates which column name of rowData to query gene.
featureDisplay	Indicates which column name of rowData to use to display feature for visualization.

Value

getBiomarker(): A data.frame of expression values

Examples

```
getBiomarker(mouseBrainSubsetSCE, gene="C1qa")
```

---

getDEGTopTable *Get Top Table of a DEG analysis*

Description

Users have to run `runDEAnalysis()` first, any of the wrapped functions of this generic function. Users can set further filters on the result. A `data.frame` object, with variables of `Gene`, `Log2_FC`, `Pvalue`, and `FDR`, will be returned.

Usage

```
getDEGTopTable(
  inSCE,
  useResult,
  labelBy = NULL,
  onlyPos = FALSE,
  log2fcThreshold = 0.25,
  fdrThreshold = 0.05
)
```

Arguments

inSCE	SingleCellExperiment inherited object, with of the singleCellTK DEG method performed in advance.
useResult	character. A string specifying the analysisName used when running a differential expression analysis function.
labelBy	A single character for a column of rowData(inSCE) as where to search for the labeling text. Default NULL for the "rownames".
onlyPos	logical. Whether to only fetch DEG with positive log2_FC value. Default FALSE.
log2fcThreshold	numeric. Only fetch DEGs with the absolute values of log2FC larger than this value. Default 0.25.
fdrThreshold	numeric. Only fetch DEGs with FDR value smaller than this value. Default 0.05.

Value

A data.frame object of the top DEGs, with variables of Gene, Log2_FC, Pvalue, and FDR.

Examples

```
data("sceBatches")
sceBatches <- scaterlogNormCounts(sceBatches, "logcounts")
sce.w <- subsetSCECols(sceBatches, colData = "batch == 'w'")
sce.w <- runWilcox(sce.w, class = "cell_type", classGroup1 = "alpha",
                   groupName1 = "w.alpha", groupName2 = "w.beta",
                   analysisName = "w.aVSb")
getDEGTopTable(sce.w, "w.aVSb")
```

---

getMSigDBTable *Shows MSigDB categories*

Description

Returns a data.frame that shows MSigDB categories and subcategories as well as descriptions for each. The entries in the ID column in this table can be used as input for importGeneSetsFromMSigDB.

Usage

```
getMSigDBTable()
```

Value

data.frame, containing MSigDB categories

Author(s)

Joshua D. Campbell

See Also

importGeneSetsFromMSigDB for importing MSigDB gene sets.

Examples

```
getMSigDBTable()
```

`getSceParams` *Extract QC parameters from the SingleCellExperiment object*

Description

Extract QC parameters from the SingleCellExperiment object

Usage

```
getSceParams(
  inSCE,
  skip = c("scrublet", "runDecontX", "runBarcodeRanksMetaOutput"),
  ignore = c("algorithms", "estimates", "contamination", "z", "sample", "rank",
    "BPPARAM", "batch", "geneSetCollection", "barcodeArgs"),
  directory = "./",
  samplename = "",
  writeYAML = TRUE
)
```

Arguments

`inSCE`	A SingleCellExperiment object.
`skip`	Skip extracting the parameters of the provided QC functions.
`ignore`	Skip extracting the content within QC functions.
`directory`	The output directory of the SCTK_runQC.R pipeline.
`samplename`	The sample name of the SingleCellExperiment objects.
`writeYAML`	Whether output yaml file to store parameters. Default if TRUE. If FALSE, return character object.

Value

If `writeYAML` TRUE, a yaml object will be generated. If FALSE, character object.

getTopHVG	*getTopHVG Extracts the top variable genes from an input* SingleCellExperiment *object. Note that the variability metrics must be computed using the 'runFeatureSelection' method before extracting the feature names of the top variable features. If 'altExp' parameter is a* character *value, this function will return the input* SingleCellExperiment *object with the subset containing only the top variable features stored as an* altExp *slot in returned object. However, if this parameter is set to* NULL*, only the names of the top variable features will be returned as a* character *vector.*

Description

getTopHVG Extracts the top variable genes from an input SingleCellExperiment object. Note that the variability metrics must be computed using the 'runFeatureSelection' method before extracting the feature names of the top variable features. If 'altExp' parameter is a character value, this function will return the input SingleCellExperiment object with the subset containing only the top variable features stored as an altExp slot in returned object. However, if this parameter is set to NULL, only the names of the top variable features will be returned as a character vector.

Usage

```
getTopHVG(inSCE, method, n = 2000, altExp = NULL)
```

Arguments

inSCE	Input SingleCellExperiment object
method	Specify which method to use for variable gene extraction from either Seurat "vst", "mean.var.plot", "dispersion" or Scran "modelGeneVar".
n	Specify the number of top variable genes to extract.
altExp	A character value that specifies the name of the altExp slot that should be created to store the subset SingleCellExperiment object containing only the top 'n' variable features. Default value is NULL, which will not store the subset SingleCellExperiment object and instead will only return the names of the top 'n' variable features.

Value

A character vector of the top variable feature names or the input SingleCellExperiment object with subset of variable features stored as an altExp in the object.

Author(s)

Irzam Sarfraz

Examples

```
data(sce_chcl, package = "scds")
sce_chcl <- scranModelGeneVar(sce_chcl, "counts")
# return top 10 variable genes
topGenes <- getTopHVG(sce_chcl, "modelGeneVar", 10)
```

`getTSNE` *Run t-SNE dimensionality reduction method on a SingleCellExperiment Object*

Description

Run t-SNE dimensionality reduction method on a SingleCellExperiment Object

Usage

```
getTSNE(
  inSCE,
  useAssay = "logcounts",
  useAltExp = NULL,
  useReducedDim = NULL,
  reducedDimName = "TSNE",
  nIterations = 1000,
  perplexity = 30,
  run_pca = TRUE,
  ntop = NULL
)
```

Arguments

`inSCE`	Input SingleCellExperiment object.
`useAssay`	Assay to use for tSNE computation. If `useAltExp` is specified, `useAssay` has to exist in `assays(altExp(inSCE,useAltExp))`. Default `"logcounts"`
`useAltExp`	The subset to use for tSNE computation, usually for the selected.variable features. Default `NULL`.
`useReducedDim`	The low dimension representation to use for UMAP computation. Default `NULL`.
`reducedDimName`	a name to store the results of the dimension reductions. Default `"TSNE"`.
`nIterations`	maximum iterations. Default `1000`.
`perplexity`	perplexity parameter. Default `30`.
`run_pca`	run tSNE on PCA components? Default `TRUE`.
`ntop`	Number of top features to use as a further variable feature selection. Default `NULL`.

Value

A SingleCellExperiment object with tSNE computation updated in `reducedDim(inSCE,reducedDimName)`.

Examples

```
data("mouseBrainSubsetSCE")
#add a CPM assay
assay(mouseBrainSubsetSCE, "cpm") <- apply(
  assay(mouseBrainSubsetSCE, "counts"), 2, function(x) {
    x / (sum(x) / 1000000)
  })
mouseBrainSubsetSCE <- getTSNE(mouseBrainSubsetSCE, useAssay = "cpm",
                               reducedDimName = "TSNE_cpm",
                               perplexity = NULL)
```

`getUMAP` *Uniform Manifold Approximation and Projection(UMAP) algorithm for dimension reduction.*

Description

Uniform Manifold Approximation and Projection(UMAP) algorithm for dimension reduction.

Usage

```
getUMAP(
  inSCE,
  useAssay = "counts",
  useAltExp = NULL,
  useReducedDim = NULL,
  sample = NULL,
  reducedDimName = "UMAP",
  logNorm = TRUE,
  nNeighbors = 30,
  nIterations = 200,
  alpha = 1,
  minDist = 0.5,
  spread = 5,
  pca = TRUE,
  initialDims = 50,
  nTop = 2000
)
```

Arguments

`inSCE`	Input SingleCellExperiment object.
`useAssay`	Assay to use for UMAP computation. If `useAltExp` is specified, `useAssay` has to exist in `assays(altExp(inSCE,useAltExp))`. Default `"counts"`.
`useAltExp`	The subset to use for UMAP computation, usually for the selected.variable features. Default `NULL`.
`useReducedDim`	The low dimension representation to use for UMAP computation. Default `NULL`.

sample	Character vector. Indicates which sample each cell belongs to. If given a single character, will take the annotation from colData. Default NULL.
reducedDimName	A name to store the results of the dimension reduction coordinates obtained from this method. Default "UMAP".
logNorm	Whether the counts will need to be log-normalized prior to generating the UMAP via logNormCounts. Will not normalize when using useReducedDim. Default TRUE.
nNeighbors	The size of local neighborhood used for manifold approximation. Larger values result in more global views of the manifold, while smaller values result in more local data being preserved. Default 30. See '?scater::calculateUMAP' for more information.
nIterations	The number of iterations performed during layout optimization. Default is 200.
alpha	The initial value of "learning rate" of layout optimization. Default is 1.
minDist	The effective minimum distance between embedded points. Smaller values will result in a more clustered/clumped embedding where nearby points on the manifold are drawn closer together, while larger values will result on a more even dispersal of points. Default 0.5. See '?scater::calculateUMAP' for more information.
spread	The effective scale of embedded points. In combination with minDist, this determines how clustered/clumped the embedded points are. Default 5. See '?scater::calculateUMAP' for more information.
pca	Logical. Whether to perform dimension reduction with PCA before UMAP. Will not perform PCA if using useReducedDim. Default TRUE
initialDims	Number of dimensions from PCA to use as input in UMAP. Default 50.
nTop	Number of features with the highest variances to use for dimensionality reduction. Default 2000.

Value

A SingleCellExperiment object with UMAP computation updated in reducedDim(inSCE,reducedDimName).

Examples

```
data(scExample, package = "singleCellTK")
sce <- subsetSCECols(sce, colData = "type != 'EmptyDroplet'")
sce <- getUMAP(inSCE = sce, useAssay = "counts", reducedDimName = "UMAP")
```

---

importAlevin *Construct SCE object from Salmon-Alevin output*

Description

Construct SCE object from Salmon-Alevin output

Usage

```
importAlevin(
  alevinDir = NULL,
  sampleName = "sample",
  delayedArray = FALSE,
  class = c("Matrix", "matrix")
)
```

Arguments

alevinDir	Character. The output directory of salmon-Alevin pipeline. It should contain subfolder named 'alevin', which contains the count data which is stored in 'quants_mat.gz'. Default NULL.
sampleName	Character. A user-defined sample name for the sample to be imported. The 'sampleName' will be appended to the begining of cell barcodes. Default is 'sample'.
delayedArray	Boolean. Whether to read the expression matrix as DelayedArray object or not. Default FALSE.
class	Character. The class of the expression matrix stored in the SCE object. Can be one of "Matrix" (as returned by readMM function), or "matrix" (as returned by matrix function). Default "Matrix".

Value

A SingleCellExperiment object containing the count matrix, the feature annotations, and the cell annotation (which includes QC metrics stored in 'featureDump.txt').

---

importAnnData *Create a SingleCellExperiment Object from Python AnnData .h5ad files*

Description

This function reads in one or more Python AnnData files in the .h5ad format and returns a single SingleCellExperiment object containing all the AnnData samples by concatenating their counts matrices and related information slots.

Usage

```
importAnnData(
  sampleDirs = NULL,
  sampleNames = NULL,
  delayedArray = FALSE,
  class = c("Matrix", "matrix")
)
```

Arguments

sampleDirs	Folder containing the .h5ad file. Can be one of - • Default current working directory. • Full path to the directory containing the .h5ad file. E.g sampleDirs = '/path/to/sample' • A vector of folder paths for the samples to import. E.g. sampleDirs = c('/path/to/sample1','/path/to/sample2','/path/to/sample3') importAnnData will return a single SCE object containing all the samples with the sample name appended to each colname in colData
sampleNames	The prefix/name of the .h5ad file without the .h5ad extension e.g. if 'sample.h5ad' is the filename, pass sampleNames = 'sample'. Can be one of - • Default sample. • A vector of samples to import. Length of vector must be equal to length of sampleDirs vector E.g. sampleDirs = c('sample1','sample2','sample3') importAnnData will return a single SCE object containing all the samples with the sample name appended to each colname in colData
delayedArray	Boolean. Whether to read the expression matrix as DelayedArray object. Default FALSE.
class	Character. The class of the expression matrix stored in the SCE object. Can be one of "Matrix" (as returned by readMM function), or "matrix" (as returned by matrix function). Default "Matrix".

Details

importAnnData converts scRNA-seq data in the AnnData format to the SingleCellExperiment object. The .X slot in AnnData is transposed to the features x cells format and becomes the 'counts' matrix in the assay slot. The .vars AnnData slot becomes the SCE rowData and the .obs AnnData slot becomes the SCE colData. Multidimensional data in the .obsm AnnData slot is ported over to the SCE reducedDims slot. Additionally, unstructured data in the .uns AnnData slot is available through the SCE metadata slot. There are 2 currently known minor issues - Anndata python module depends on another python module h5pyto read hd5 format files. If there are errors reading the .h5ad files, such as "ValueError: invalid shape in fixed-type tuple." the user will need to do downgrade h5py by running pip3 install --user h5py==2.9.0 Additionally there might be errors in converting some python objects in the unstructured data slots. There are no known R solutions at present. Refer https://github.com/rstudio/reticulate/issues/209

Value

A SingleCellExperiment object.

Examples

```
file.path <- system.file("extdata/annData_pbmc_3k", package = "singleCellTK")
## Not run:
sce <- importAnnData(sampleDirs = file.path,
                     sampleNames = 'pbmc3k_20by20')

## End(Not run)
```

importBUStools *Construct SCE object from BUStools output*

Description

Read the barcodes, features (genes), and matrix from BUStools output. Import them as one SingleCellExperiment object. Note the cells in the output files for BUStools 0.39.4 are not filtered.

Usage

```
importBUStools(
  BUStoolsDirs,
  samples,
  matrixFileNames = "genes.mtx",
  featuresFileNames = "genes.genes.txt",
  barcodesFileNames = "genes.barcodes.txt",
  gzipped = "auto",
  class = c("Matrix", "matrix"),
  delayedArray = FALSE
)
```

Arguments

BUStoolsDirs	A vector of paths to BUStools output files. Each sample should have its own path. For example: `./genecount`. Must have the same length as `samples`.
samples	A vector of user-defined sample names for the samples to be imported. Must have the same length as `BUStoolsDirs`.
matrixFileNames	Filenames for the Market Exchange Format (MEX) sparse matrix files (.mtx files). Must have length 1 or the same length as `samples`.
featuresFileNames	Filenames for the feature annotation files. Must have length 1 or the same length as `samples`.
barcodesFileNames	Filenames for the cell barcode list file. Must have length 1 or the same length as `samples`.
gzipped	Boolean. `TRUE` if the BUStools output files (barcodes.txt, genes.txt, and genes.mtx) were gzip compressed. `FALSE` otherwise. This is `FALSE` in BUStools 0.39.4. Default `"auto"` which automatically detects if the files are gzip compressed. Must have length 1 or the same length as `samples`.
class	Character. The class of the expression matrix stored in the SCE object. Can be one of "Matrix" (as returned by readMM function), or "matrix" (as returned by matrix function). Default "Matrix".
delayedArray	Boolean. Whether to read the expression matrix as DelayedArray-class object or not. Default `FALSE`.

Value

A SingleCellExperiment object containing the count matrix, the gene annotation, and the cell annotation.

Examples

```
# Example #1
# FASTQ files were downloaded from
# https://support.10xgenomics.com/single-cell-gene-expression/datasets/3.0.0
# /pbmc_1k_v3
# They were concatenated as follows:
# cat pbmc_1k_v3_S1_L001_R1_001.fastq.gz pbmc_1k_v3_S1_L002_R1_001.fastq.gz >
# pbmc_1k_v3_R1.fastq.gz
# cat pbmc_1k_v3_S1_L001_R2_001.fastq.gz pbmc_1k_v3_S1_L002_R2_001.fastq.gz >
# pbmc_1k_v3_R2.fastq.gz
# The following BUStools command generates the gene, cell, and
# matrix files

# bustools correct -w ./3M-february-2018.txt -p output.bus | \
#   bustools sort -T tmp/ -t 4 -p - | \
#   bustools count -o genecount/genes \
#     -g ./transcripts_to_genes.txt \
#     -e matrix.ec \
#     -t transcripts.txt \
#     --genecounts -

# The top 20 genes and the first 20 cells are included in this example.
sce <- importBUStools(
  BUStoolsDirs = system.file("extdata/BUStools_PBMC_1k_v3_20x20/genecount/",
    package = "singleCellTK"),
  samples = "PBMC_1k_v3_20x20")
```

---

importCellRanger *Construct SCE object from Cell Ranger output*

Description

Read the filtered barcodes, features, and matrices for all samples from (preferably a single run of) Cell Ranger output. Import and combine them as one big SingleCellExperiment object.

Usage

```
importCellRanger(
  cellRangerDirs = NULL,
  sampleDirs = NULL,
  sampleNames = NULL,
  cellRangerOuts = NULL,
  dataType = c("filtered", "raw"),
  matrixFileNames = "matrix.mtx.gz",
```

```
  featuresFileNames = "features.tsv.gz",
  barcodesFileNames = "barcodes.tsv.gz",
  gzipped = "auto",
  class = c("Matrix", "matrix"),
  delayedArray = FALSE
)

importCellRangerV2(
  cellRangerDirs = NULL,
  sampleDirs = NULL,
  sampleNames = NULL,
  dataTypeV2 = c("filtered", "raw"),
  class = c("Matrix", "matrix"),
  delayedArray = FALSE,
  reference = NULL,
  cellRangerOutsV2 = NULL
)

importCellRangerV3(
  cellRangerDirs = NULL,
  sampleDirs = NULL,
  sampleNames = NULL,
  dataType = c("filtered", "raw"),
  class = c("Matrix", "matrix"),
  delayedArray = FALSE
)
```

Arguments

`cellRangerDirs`	The root directories where Cell Ranger was run. These folders should contain sample specific folders. Default `NULL`, meaning the paths for each sample will be specified in *samples* argument.
`sampleDirs`	Default `NULL`. Can be one of • `NULL`. All samples within `cellRangerDirs` will be imported. The order of samples will be first determined by the order of `cellRangerDirs` and then by list.dirs. This is only for the case where `cellRangerDirs` is specified. • A list of vectors containing the folder names for samples to import. Each vector in the list corresponds to samples from one of `cellRangerDirs`. These names are the same as the folder names under `cellRangerDirs`. This is only for the case where `cellRangerDirs` is specified. • A vector of folder paths for the samples to import. This is only for the case where `cellRangerDirs` is `NULL`. The cells in the final SCE object will be ordered in the same order of `sampleDirs`.
`sampleNames`	A vector of user-defined sample names for the samples to be imported. Must have the same length as `length(unlist(sampleDirs))` if `sampleDirs` is not `NULL`. Otherwise, make sure the length and order match the output of `unlist(lapply(cellRangerDirs,l = FALSE))`. Default `NULL`, in which case the folder names will be used as sample names.

`cellRangerOuts` Character vector. The intermediate paths to filtered or raw cell barcode, feature, and matrix files for each sample. **Supercedes** `dayaType`. If `NULL`, `dataType` will be used to determine Cell Ranger output directory. If not `NULL`, `dataType` will be ingored and `cellRangerOuts` specifies the paths. Must have length 1 or the same length as `length(unlist(sampleDirs))` if `sampleDirs` is not `NULL`. Otherwise, make sure the length and order match the output of `unlist(lapply(cellRangerDirs,list.dirs = FALSE))`. Reference genome names might need to be appended for CellRanger version below 3.0.0 if reads were mapped to multiple genomes when running Cell Ranger pipeline. Probable options include "outs/filtered_feature_bc_matrix/", "outs/raw_feature_bc_matrix/", "outs/filtered_gene_bc_matrix/", "outs/raw_gene_bc_matrix/".

`dataType` Character. The type of data to import. Can be one of "filtered" (which is equivalent to `cellRangerOuts = "outs/filtered_feature_bc_matrix/"` or `cellRangerOuts = "outs/filtered_gene_bc_matrix/"`) or "raw" (which is equivalent to `cellRangerOuts = "outs/raw_feature_bc_matrix/"` or `cellRangerOuts = "outs/raw_gene_bc_matrix/"`). Default "filtered" which imports the counts for filtered cell barcodes only.

`matrixFileNames` Character vector. Filenames for the Market Exchange Format (MEX) sparse matrix files (matrix.mtx or matrix.mtx.gz files). Must have length 1 or the same length as `length(unlist(sampleDirs))` if `sampleDirs` is not `NULL`. Otherwise, make sure the length and order match the output of `unlist(lapply(cellRangerDirs,list.dirs, = FALSE))`.

`featuresFileNames` Character vector. Filenames for the feature annotation files. They are usually named *features.tsv.gz* or *genes.tsv*. Must have length 1 or the same length as `length(unlist(sampleDirs))` if `sampleDirs` is not `NULL`. Otherwise, make sure the length and order match the output of `unlist(lapply(cellRangerDirs,list.dirs,recursive = FALSE))`.

`barcodesFileNames` Character vector. Filename for the cell barcode list files. They are usually named *barcodes.tsv.gz* or *barcodes.tsv*. Must have length 1 or the same length as `length(unlist(sampleDirs))` if `sampleDirs` is not `NULL`. Otherwise, make sure the length and order match the output of `unlist(lapply(cellRangerDirs,list.dirs,recursive = FALSE))`.

`gzipped` `TRUE` if the Cell Ranger output files (barcodes.tsv, features.tsv, and matrix.mtx) were gzip compressed. `FALSE` otherwise. This is true after Cell Ranger 3.0.0 update. Default `"auto"` which automatically detects if the files are gzip compressed. If not `"auto"`, `gzipped` must have length 1 or the same length as `length(unlist(sampleDirs))` if `sampleDirs` is not `NULL`. Otherwise, make sure the length and order match the output of `unlist(lapply(cellRangerDirs,list.dirs,recursive = FALSE))`.

`class` Character. The class of the expression matrix stored in the SCE object. Can be one of "Matrix" (as returned by readMM function), or "matrix" (as returned by matrix function). Default `"Matrix"`.

`delayedArray` Boolean. Whether to read the expression matrix as DelayedArray object or not. Default `FALSE`.

dataTypeV2 Character. The type of output to import for Cellranger version below 3.0.0. Whether to import the filtered or the raw data. Can be one of 'filtered' or 'raw'. Default 'filtered'. When cellRangerOuts is specified, dataTypeV2 and reference will be ignored.

reference Character vector. The reference genome names. Default NULL. If not NULL, it must gave the length and order as length(unlist(sampleDirs)) if sampleDirs is not NULL. Otherwise, make sure the length and order match the output of unlist(lapply(cellRangerDirs,list.dirs,recursive = FALSE)). Only needed for Cellranger version below 3.0.0.

cellRangerOutsV2 Character vector. The intermediate paths to filtered or raw cell barcode, feature, and matrix files for each sample for Cellranger version below 3.0.0. If NULL, reference and dataTypeV2 will be used to determine Cell Ranger output directory. If it has length 1, it assumes that all samples use the same genome reference and the function will load only filtered or raw data.

Details

importCellRangerV2 imports output from Cell Ranger V2. importCellRangerV2Sample imports output from one sample from Cell Ranger V2. importCellRangerV3 imports output from Cell Ranger V3. importCellRangerV3 imports output from one sample from Cell Ranger V3. Some implicit assumptions which match the output structure of Cell Ranger V2 & V3 are made in these 4 functions including cellRangerOuts, matrixFileName, featuresFileName, barcodesFileName, and gzipped. Alternatively, user can call importCellRanger to explicitly specify these arguments.

Value

A SingleCellExperiment object containing the combined count matrix, the feature annotations, and the cell annotation.

Examples

```
# Example #1
# The following filtered feature, cell, and matrix files were downloaded from
# https://support.10xgenomics.com/single-cell-gene-expression/datasets/
# 3.0.0/hgmm_1k_v3
# The top 10 hg19 & mm10 genes are included in this example.
# Only the first 20 cells are included.
sce <- importCellRanger(
    cellRangerDirs = system.file("extdata/", package = "singleCellTK"),
    sampleDirs = "hgmm_1k_v3_20x20",
    sampleNames = "hgmm1kv3",
    dataType = "filtered")
# The following filtered feature, cell, and matrix files were downloaded from
# https://support.10xgenomics.com/single-cell-gene-expression/datasets/
# 2.1.0/pbmc4k
# Top 20 genes are kept. 20 cell barcodes are extracted.
sce <- importCellRangerV2(
    cellRangerDirs = system.file("extdata/", package = "singleCellTK"),
    sampleDirs = "pbmc_4k_v2_20x20",
```

```
    sampleNames = "pbmc4k_20",
    reference = 'GRCh38',
    dataTypeV2 = "filtered")
sce <- importCellRangerV3(
    cellRangerDirs = system.file("extdata/", package = "singleCellTK"),
    sampleDirs = "hgmm_1k_v3_20x20",
    sampleNames = "hgmm1kv3",
    dataType = "filtered")
```

`importCellRangerV2Sample` *Construct SCE object from Cell Ranger V2 output for a single sample*

Description

Read the filtered barcodes, features, and matrices for all samples from Cell Ranger V2 output. Files are assumed to be named "matrix.mtx", "genes.tsv", and "barcodes.tsv".

Usage

```
importCellRangerV2Sample(
  dataDir = NULL,
  sampleName = NULL,
  class = c("Matrix", "matrix"),
  delayedArray = FALSE
)
```

Arguments

`dataDir`	A path to the directory containing the data files. Default "./".
`sampleName`	A User-defined sample name. This will be prepended to all cell barcode IDs. Default "sample".
`class`	Character. The class of the expression matrix stored in the SCE object. Can be one of "Matrix" (as returned by readMM function), or "matrix" (as returned by matrix function). Default "Matrix".
`delayedArray`	Boolean. Whether to read the expression matrix as DelayedArray object or not. Default `FALSE`.

Value

A `SingleCellExperiment` object containing the count matrix, the feature annotations, and the cell annotation for the sample.

Examples

```
sce <- importCellRangerV2Sample(
    dataDir = system.file("extdata/pbmc_4k_v2_20x20/outs/",
        "filtered_gene_bc_matrices/GRCh38", package = "singleCellTK"),
    sampleName = "pbmc4k_20")
```

`importCellRangerV3Sample` *Construct SCE object from Cell Ranger V3 output for a single sample*

Description

Read the filtered barcodes, features, and matrices for all samples from Cell Ranger V3 output. Files are assumed to be named "matrix.mtx.gz", "features.tsv.gz", and "barcodes.tsv.gz".

Usage

```
importCellRangerV3Sample(
  dataDir = "./",
  sampleName = "sample",
  class = c("Matrix", "matrix"),
  delayedArray = FALSE
)
```

Arguments

`dataDir`	A path to the directory containing the data files. Default "./".
`sampleName`	A User-defined sample name. This will be prepended to all cell barcode IDs. Default "sample".
`class`	Character. The class of the expression matrix stored in the SCE object. Can be one of "Matrix" (as returned by readMM function), or "matrix" (as returned by matrix function). Default "Matrix".
`delayedArray`	Boolean. Whether to read the expression matrix as DelayedArray object or not. Default `FALSE`.

Value

A `SingleCellExperiment` object containing the count matrix, the feature annotations, and the cell annotation for the sample.

Examples

```
sce <- importCellRangerV3Sample(
    dataDir = system.file("extdata/hgmm_1k_v3_20x20/outs/",
        "filtered_feature_bc_matrix", package = "singleCellTK"),
    sampleName = "hgmm1kv3")
```

`importDropEst` *Create a SingleCellExperiment Object from DropEst output*

Description

imports the RDS file created by DropEst (https://github.com/hms-dbmi/dropEst) and create a SingleCellExperiment object from either the raw or filtered counts matrix. Additionally parse through the RDS to obtain appropriate feature annotations as SCE coldata, in addition to any metadata.

Usage

```
importDropEst(
  sampleDirs = NULL,
  dataType = c("filtered", "raw"),
  rdsFileName = "cell.counts",
  sampleNames = NULL,
  delayedArray = FALSE,
  class = c("Matrix", "matrix")
)
```

Arguments

`sampleDirs`	A path to the directory containing the data files. Default "./".
`dataType`	can be "filtered" or "raw". Default `"filtered"`.
`rdsFileName`	File name prefix of the DropEst RDS output. default is "cell.counts"
`sampleNames`	A User-defined sample name. This will be prepended to all cell barcode IDs. Default "sample".
`delayedArray`	Boolean. Whether to read the expression matrix as DelayedArray object or not. Default `FALSE`.
`class`	Character. The class of the expression matrix stored in the SCE object. Can be one of "Matrix" (as returned by readMM function), or "matrix" (as returned by matrix function). Default `"Matrix"`.

Details

`importDropEst` expects either raw counts matrix stored as "cm_raw" or filtered counts matrix stored as "cm" in the DropEst rds output. ColData is obtained from the DropEst corresponding to "mean_reads_per_umi","aligned_reads_per_cell", "aligned_umis_per_cell","requested_umis_per_cb","requested_reads_per
If using filtered counts matrix, the colData dataframe is subset to contain features from the filtered counts matrix alone. If any annotations of ("saturation_info","merge_targets","reads_per_umi_per_cell") are found in the DropEst rds, they will be added to the SCE metadata field

Value

A `SingleCellExperiment` object containing the count matrix, the feature annotations from DropEst as ColData, and any metadata from DropEst

Examples

```
# Example results were generated as per instructions from the developers of dropEst described in
# https://github.com/hms-dbmi/dropEst/blob/master/examples/EXAMPLES.md
sce <- importDropEst(sampleDirs = system.file("extdata/dropEst_scg71", package = "singleCellTK"),
                     sampleNames = 'scg71')
```

`importExampleData` *Retrieve example datasets*

Description

Retrieves published example datasets stored in SingleCellExperiment using the scRNAseq and TENxPBMCData packages. See 'Details' for a list of available datasets.

Usage

```
importExampleData(dataset, class = c("Matrix", "matrix"), delayedArray = FALSE)
```

Arguments

`dataset`	Character. Name of the dataset to retrieve.
`class`	Character. The class of the expression matrix stored in the SCE object. Can be one of `"Matrix"` or `"matrix"`. `"Matrix"` will store the data as a sparse matrix from package Matrix while `"matrix"` will store the data in a standard matrix. Default `"Matrix"`.
`delayedArray`	Boolean. Whether to read the expression matrix as DelayedArray object or not. Default `FALSE`.

Details

See the list below for the available datasets and their descriptions.

"fluidigm_pollen" Retrieved with `ReprocessedFluidigmData`. Returns a dataset of 65 human neural cells from Pollen et al. (2014), each sequenced at high and low coverage (SRA accession SRP041736).

"allen_tasic" Retrieved with `ReprocessedAllenData`. Returns a dataset of 379 mouse brain cells from Tasic et al. (2016).

"pbmc3k" Retrieved with `TENxPBMCData`. 2,700 peripheral blood mononuclear cells (PBMCs) from 10X Genomics.

"pbmc4k" Retrieved with `TENxPBMCData`. 4,340 peripheral blood mononuclear cells (PBMCs) from 10X Genomics.

"pbmc6k" Retrieved with `TENxPBMCData`. 5,419 peripheral blood mononuclear cells (PBMCs) from 10X Genomics.

"pbmc8k" Retrieved with `TENxPBMCData`. 8,381 peripheral blood mononuclear cells (PBMCs) from 10X Genomics.

"pbmc33k" Retrieved with TENxPBMCData. 33,148 peripheral blood mononuclear cells (PBMCs) from 10X Genomics.

"pbmc68k" Retrieved with TENxPBMCData. 68,579 peripheral blood mononuclear cells (PBMCs) from 10X Genomics.

Value

The specified SingleCellExperiment object.

Author(s)

Joshua D. Campbell, David Jenkins

Examples

```
sce <- importExampleData("pbmc3k")
```

---

importFromFiles *Create a SingleCellExperiment object from files*

Description

Creates a SingleCellExperiment object from a counts file in various formats. and a file of annotation information, .

Usage

```
importFromFiles(
  assayFile,
  annotFile = NULL,
  featureFile = NULL,
  assayName = "counts",
  inputDataFrames = FALSE,
  class = c("Matrix", "matrix"),
  delayedArray = FALSE,
  annotFileHeader = FALSE,
  annotFileRowName = 1,
  annotFileSep = "\t",
  featureHeader = FALSE,
  featureRowName = 1,
  featureSep = "\t",
  gzipped = "auto"
)
```

Arguments

assayFile	The path to a file in .mtx, .txt, .csv, .tab, or .tsv format.
annotFile	The path to a text file that contains columns of annotation information for each sample in the assayFile. This file should have the same number of rows as there are columns in the assayFile. If multiple samples are represented in these files, this should be denoted by a column called 'sample' within the annotFile.
featureFile	The path to a text file that contains columns of annotation information for each gene in the count matrix. This file should have the same genes in the same order as assayFile. This is optional.
assayName	The name of the assay that you are uploading. The default is "counts".
inputDataFrames	If TRUE, assayFile and annotFile are read as data frames instead of file paths. The default is FALSE.
class	Character. The class of the expression matrix stored in the SCE object. Can be one of "Matrix" (as returned by readMM function), or "matrix" (as returned by matrix function). Default "Matrix".
delayedArray	Boolean. Whether to read the expression matrix as DelayedArray object or not. Default FALSE.
annotFileHeader	Whether there's a header (colnames) in the cell annotation file. Default is FALSE
annotFileRowName	Which column is used as the rownames for the cell annotation file. Default is 1 (first column).
annotFileSep	Separater used for the cell annotation file. Default is "\t".
featureHeader	Whether there's a header (colnames) in the feature annotation file. Default is FALSE
featureRowName	Which column is used as the rownames for the feature annotation file. Default is 1 (first column).
featureSep	Separater used for the feature annotation file. Default is "\t".
gzipped	Whether the input file is gzipped. Default is "auto" and it will automatically detect whether the file is gzipped. Other options is TRUE or FALSE.

Value

a SingleCellExperiment object

`importGeneSetsFromCollection` *Imports gene sets from a GeneSetCollection object*

Description

Converts a list of gene sets stored in a GeneSetCollection object and stores it in the metadata of the SingleCellExperiment object. These gene sets can be used in downstream quality control and analysis functions in singleCellTK.

Usage

```
importGeneSetsFromCollection(
  inSCE,
  geneSetCollection,
  collectionName = "GeneSetCollection",
  by = "rownames"
)
```

Arguments

inSCE	Input SingleCellExperiment object.
geneSetCollection	A GeneSetCollection object. See GeneSetCollection for more details.
collectionName	Character. Name of collection to add gene sets to. If this collection already exists in inSCE, then these gene sets will be added to that collection. Any gene sets within the collection with the same name will be overwritten. Default GeneSetCollection.
by	Character, character vector, or NULL. Describes the location within inSCE where the gene identifiers in geneSetCollection should be mapped. If set to "rownames" then the features will be searched for among rownames(inSCE). This can also be set to one of the column names of rowData(inSCE) in which case the gene identifies will be mapped to that column in the rowData of inSCE. by can be a vector the same length as the number of gene sets in the GeneSetCollection and the elements of the vector can point to different locations within inSCE. Finally, by can be NULL. In this case, the location of the gene identifiers in inSCE should be saved in the description slot for each gene set in the GeneSetCollection. See featureIndex for more information. Default "rownames".

Details

The gene identifiers in gene sets in the GeneSetCollection will be mapped to the rownames of inSCE using the by parameter and stored in a GeneSetCollection object from package GSEABase. This object is stored in metadata(inSCE)$sctk$genesets, which can be accessed in downstream analysis functions such as runCellQC.

Value

A SingleCellExperiment object with gene set from collectionName output stored to the metadata slot.

Author(s)

Joshua D. Campbell

See Also

importGeneSetsFromList for importing from lists, importGeneSetsFromGMT for importing from GMT files, and importGeneSetsFromMSigDB for importing MSigDB gene sets.

Examples

```
data(scExample)
library(GSEABase)
gs1 <- GeneSet(setName = "geneset1", geneIds = rownames(sce)[seq(10)])
gs2 <- GeneSet(setName = "geneset2", geneIds = rownames(sce)[seq(11,20)])
gsc <- GeneSetCollection(list(gs1, gs2))
sce <- importGeneSetsFromCollection(inSCE = sce,
                                    geneSetCollection = gsc,
                                    by = "rownames")
```

`importGeneSetsFromGMT` *Imports gene sets from a GMT file*

Description

Converts a list of gene sets stored in a GMT file into a GeneSetCollection and stores it in the metadata of the SingleCellExperiment object. These gene sets can be used in downstream quality control and analysis functions in singleCellTK.

Usage

```
importGeneSetsFromGMT(
  inSCE,
  file,
  collectionName = "GeneSetCollection",
  by = "rownames",
  sep = "\t"
)
```

Arguments

`inSCE`	Input SingleCellExperiment object.
`file`	Character. Path to GMT file. See getGmt for more information on reading GMT files.
`collectionName`	Character. Name of collection to add gene sets to. If this collection already exists in `inSCE`, then these gene sets will be added to that collection. Any gene sets within the collection with the same name will be overwritten. Default `GeneSetCollection`.
`by`	Character, character vector, or NULL. Describes the location within `inSCE` where the gene identifiers in `geneSetList` should be mapped. If set to `"rownames"` then the features will be searched for among `rownames(inSCE)`. This can also be set to one of the column names of `rowData(inSCE)` in which case the gene identifies will be mapped to that column in the `rowData` of `inSCE`. `by` can be a vector the same length as the number of gene sets in the GMT file and the elements of the vector can point to different locations within `inSCE`. Finally, `by` can be `NULL`. In this case, the location of the gene identifiers in `inSCE` should be saved in the description (2nd column) of the GMT file. See featureIndex for more information. Default `"rownames"`.

sep Character. Delimiter of the GMT file. Default "\t".

Details

The gene identifiers in gene sets in the GMT file will be mapped to the rownames of inSCE using the by parameter and stored in a GeneSetCollection object from package GSEABase. This object is stored in metadata(inSCE)$sctk$genesets, which can be accessed in downstream analysis functions such as runCellQC.

Value

A SingleCellExperiment object with gene set from collectionName output stored to the metadata slot.

Author(s)

Joshua D. Campbell

See Also

importGeneSetsFromList for importing from lists, importGeneSetsFromCollection for importing from GeneSetCollection objects, and importGeneSetsFromMSigDB for importing MSigDB gene sets.

Examples

```
data(scExample)

# GMT file containing gene symbols for a subset of human mitochondrial genes
gmt <- system.file("extdata/mito_subset.gmt", package = "singleCellTK")

# "feature_name" is the second column in the GMT file, so the ids will
# be mapped using this column in the 'rowData' of 'sce'. This
# could also be accomplished by setting by = "feature_name" in the
# function call.
sce <- importGeneSetsFromGMT(inSCE = sce, file = gmt, by = NULL)
```

---

importGeneSetsFromList

Imports gene sets from a list

Description

Converts a list of gene sets into a GeneSetCollection and stores it in the metadata of the SingleCellExperiment object. These gene sets can be used in downstream quality control and analysis functions in singleCellTK.

Usage

```
importGeneSetsFromList(
  inSCE,
  geneSetList,
  collectionName = "GeneSetCollection",
  by = "rownames"
)
```

Arguments

inSCE	Input SingleCellExperiment object.
geneSetList	Named List. A list containing one or more gene sets. Each element of the list should be a character vector of gene identifiers. The names of the list will be become the gene set names in the GeneSetCollection object.
collectionName	Character. Name of collection to add gene sets to. If this collection already exists in inSCE, then these gene sets will be added to that collection. Any gene sets within the collection with the same name will be overwritten. Default GeneSetCollection.
by	Character or character vector. Describes the location within inSCE where the gene identifiers in geneSetList should be mapped. If set to "rownames" then the features will be searched for among rownames(inSCE). This can also be set to one of the column names of rowData(inSCE) in which case the gene identifies will be mapped to that column in the rowData of inSCE. Finally, by can be a vector the same length as the number of gene sets in geneSetList and the elements of the vector can point to different locations within inSCE. See featureIndex for more information. Default "rownames".

Details

The gene identifiers in gene sets in geneSetList will be mapped to the rownames of inSCE using the by parameter and stored in a GeneSetCollection object from package GSEABase. This object is stored in metadata(inSCE)$sctk$genesets, which can be accessed in downstream analysis functions such as runCellQC.

Value

A SingleCellExperiment object with gene set from collectionName output stored to the metadata slot.

Author(s)

Joshua D. Campbell

See Also

importGeneSetsFromCollection for importing from GeneSetCollection objects, importGeneSetsFromGMT for importing from GMT files, and importGeneSetsFromMSigDB for importing MSigDB gene sets.

Examples

```
data(scExample)

# Generate gene sets from 'rownames'
gs1 <- rownames(sce)[seq(10)]
gs2 <- rownames(sce)[seq(11,20)]
gs <- list("geneset1" = gs1, "geneset2" = gs2)
sce <- importGeneSetsFromList(inSCE = sce,
                              geneSetList = gs,
                              by = "rownames")

# Generate a gene set for mitochondrial genes using
# Gene Symbols stored in 'rowData'
mito.ix <- grep("^MT-", rowData(sce)$feature_name)
mito <- list(mito = rowData(sce)$feature_name[mito.ix])
sce <- importGeneSetsFromList(inSCE = sce,
                             geneSetList = mito,
                             by = "feature_name")
```

---

importGeneSetsFromMSigDB

Imports gene sets from MSigDB

Description

Gets a list of MSigDB gene sets stores it in the metadata of the SingleCellExperiment object. These gene sets can be used in downstream quality control and analysis functions in singleCellTK.

Usage

```
importGeneSetsFromMSigDB(
  inSCE,
  categoryIDs,
  species = "Homo sapiens",
  mapping = c("gene_symbol", "human_gene_symbol", "entrez_gene"),
  by = "rownames",
  verbose = TRUE
)
```

Arguments

inSCE	Input SingleCellExperiment object.
categoryIDs	Character vector containing the MSigDB gene set ids. The column ID in the table returned by getMSigDBTable() shows the list of possible gene set IDs that can be obtained.
species	Character. Species available can be found using the function msigdbr_show_species. Default "Homo sapiens".

mapping	Character. One of "gene_symbol", "human_gene_symbol", or "entrez_gene". Gene identifiers to be used for MSigDB gene sets. IDs denoted by the by parameter must be either in gene symbol or Entrez gene id format to match IDs from MSigDB.
by	Character. Describes the location within inSCE where the gene identifiers in the MSigDB gene sets should be mapped. If set to "rownames" then the features will be searched for among rownames(inSCE). This can also be set to one of the column names of rowData(inSCE) in which case the gene identifies will be mapped to that column in the rowData of inSCE. See featureIndex for more information. Default "rownames".
verbose	Boolean. Whether to display progress. Default TRUE.

Details

The gene identifiers in gene sets from MSigDB will be retrieved using the msigdbr package. They will be mapped to the IDs in inSCE using the by parameter and stored in a GeneSetCollection object from package GSEABase. This object is stored in metadata(inSCE)$sctk$genesets, which can be accessed in downstream analysis functions such as runCellQC.

Value

A SingleCellExperiment object with gene set from collectionName output stored to the metadata slot.

Author(s)

Joshua D. Campbell

See Also

importGeneSetsFromList for importing from lists, importGeneSetsFromGMT for importing from GMT files, and GeneSetCollection objects.

Examples

```
data(scExample)
sce <- importGeneSetsFromMSigDB(inSCE = sce,
                                categoryIDs = "H",
                                species = "Homo sapiens",
                                mapping = "gene_symbol",
                                by = "feature_name")
```

`importMitoGeneSet` *Import mitochondrial gene sets*

Description

Imports mitochondrial gene sets and stores it in the metadata of the SingleCellExperiment object. These gene sets can be used in downstream quality control and analysis functions in singleCellTK.

Usage

```
importMitoGeneSet(inSCE, reference, id, by, collectionName)
```

Arguments

`inSCE`	Input SingleCellExperiment object.
`reference`	Character. Species available are "human" and "mouse".
`id`	Types of gene id. Now it supports "symbol", "entrez", "ensembl" and "ensemblTranscriptID".
`by`	Character. Describes the location within `inSCE` where the gene identifiers in the mitochondrial gene sets should be mapped. If set to `"rownames"` then the features will be searched for among `rownames(inSCE)`. This can also be set to one of the column names of `rowData(inSCE)` in which case the gene identifies will be mapped to that column in the `rowData` of `inSCE`. See featureIndex for more information. Default `"rownames"`.
`collectionName`	Character. Name of collection to add gene sets to. If this collection already exists in `inSCE`, then these gene sets will be added to that collection. Any gene sets within the collection with the same name will be overwritten.

Details

The gene identifiers of mitochondrial genes will be loaded with "data(AllMito)". Currently, it supports human and mouse reference. Also, it supports entrez ID, gene symbol, ensemble ID and ensemble transcript ID. They will be mapped to the IDs in `inSCE` using the `by` parameter and stored in a GeneSetCollection object from package GSEABase. This object is stored in `metadata(inSCE)$sctk$genesets`, which can be accessed in downstream analysis functions such as runCellQC.

Value

A SingleCellExperiment object with gene set from `collectionName` output stored to the metadata slot.

Author(s)

Rui Hong

See Also

importGeneSetsFromList for importing from lists, importGeneSetsFromGMT for importing from GMT files, and GeneSetCollection objects.

Examples

```
data(scExample)
sce <- importMitoGeneSet(inSCE = sce,
                         reference = "human",
                         id = "ensembl",
                         collectionName = "human_mito",
                         by = "rownames")
```

---

importMultipleSources *Imports samples from different sources and compiles them into a list of SCE objects*

Description

Imports samples from different sources and compiles them into a list of SCE objects

Usage

```
importMultipleSources(allImportEntries, delayedArray = FALSE)
```

Arguments

allImportEntries	object containing the sources and parameters of all the samples being imported (from the UI)
delayedArray	Boolean. Whether to read the expression matrix as DelayedArray object or not. Default FALSE.

Value

A list of SingleCellExperiment object containing the droplet or cell data or both,depending on the dataType that users provided.

`importOptimus` *Construct SCE object from Optimus output*

Description

Read the barcodes, features (genes), and matrices from Optimus outputs. Import them as one SingleCellExperiment object.

Usage

```
importOptimus(
  OptimusDirs,
  samples,
  matrixLocation = "call-MergeCountFiles/sparse_counts.npz",
  colIndexLocation = "call-MergeCountFiles/sparse_counts_col_index.npy",
  rowIndexLocation = "call-MergeCountFiles/sparse_counts_row_index.npy",
  cellMetricsLocation = "call-MergeCellMetrics/merged-cell-metrics.csv.gz",
  geneMetricsLocation = "call-MergeGeneMetrics/merged-gene-metrics.csv.gz",
  emptyDropsLocation = "call-RunEmptyDrops/empty_drops_result.csv",
  class = c("Matrix", "matrix"),
  delayedArray = FALSE
)
```

Arguments

`OptimusDirs`	A vector of root directories of Optimus output files. The paths should be something like this: `/PATH/TO/bb4a2a5e-ff34-41b6-97d2-0c0c0c534530`. Each entry in `OptimusDirs` is considered a sample and should have its own path. Must have the same length as `samples`.
`samples`	A vector of user-defined sample names for the sample to be imported. Must have the same length as `OptimusDirs`.
`matrixLocation`	Character. It is the intermediate path to the filtered count maxtrix file saved in sparse matrix format (`.npz`). Default `call-MergeCountFiles/sparse_counts.npz` which works for optimus_v1.4.0.
`colIndexLocation`	Character. The intermediate path to the barcode index file. Default `call-MergeCountFiles/sparse_cou`
`rowIndexLocation`	Character. The intermediate path to the feature (gene) index file. Default `call-MergeCountFiles/sparse`
`cellMetricsLocation`	Character. It is the intermediate path to the cell metrics file (`merged-cell-metrics.csv.gz`). Default `call-MergeCellMetrics/merged-cell-metrics.csv.gz` which works for optimus_v1.4.0.
`geneMetricsLocation`	Character. It is the intermediate path to the feature (gene) metrics file (`merged-gene-metrics.csv.gz`). Default `call-MergeGeneMetrics/merged-gene-metrics.csv.gz` which works for optimus_v1.4.0.

emptyDropsLocation
: Character. It is the intermediate path to emptyDrops metrics file (`empty_drops_result.csv`). Default `call-RunEmptyDrops/empty_drops_result.csv` which works for optimus_v1.4.0.

class
: Character. The class of the expression matrix stored in the SCE object. Can be one of "Matrix" (as returned by readMM function), or "matrix" (as returned by matrix function). Default "Matrix".

delayedArray
: Boolean. Whether to read the expression matrix as DelayedArray object or not. Default `FALSE`.

Value

A SingleCellExperiment object containing the count matrix, the gene annotation, and the cell annotation.

Examples

```
file.path <- system.file("extdata/Optimus_20x1000",
  package = "singleCellTK")
## Not run:
sce <- importOptimus(OptimusDirs = file.path,
  samples = "Optimus_20x1000")

## End(Not run)
```

---

importSEQC *Construct SCE object from seqc output*

Description

Read the filtered barcodes, features, and matrices for all samples from (preferably a single run of) seqc output. Import and combine them as one big SingleCellExperiment object.

Usage

```
importSEQC(
  seqcDirs = NULL,
  samples = NULL,
  prefix = NULL,
  gzipped = FALSE,
  class = c("Matrix", "matrix"),
  delayedArray = FALSE,
  cbNotFirstCol = TRUE,
  feNotFirstCol = TRUE,
  combinedSample = TRUE
)
```

Arguments

seqcDirs	A vector of paths to seqc output files. Each sample should have its own path. For example: ./pbmc_1k_50x50. Must have the same length as samples.
samples	A vector of user-defined sample names for the samples to be imported. Must have the same length as seqcDirs.
prefix	A vector containing the prefix of file names within each sample directory. It cannot be null and the vector should have the same length as *samples*.
gzipped	Boolean. TRUE if the seqc output files (sparse_counts_barcode.csv, sparse_counts_genes.csv, and sparse_molecule_counts.mtx) were gzip compressed. FALSE otherwise. Default seqc outputs are not gzipped. Default FALSE.
class	Character. The class of the expression matrix stored in the SCE object. Can be one of "Matrix" (as returned by readMM function), or "matrix" (as returned by matrix function). Default "Matrix".
delayedArray	Boolean. Whether to read the expression matrix as DelayedArray object or not. Default FALSE.
cbNotFirstCol	Boolean. TRUE if first column of sparse_counts_barcode.csv is row index and it will be removed. FALSE the first column will be kept.
feNotFirstCol	Boolean. TRUE if first column of sparse_counts_genes.csv is row index and it will be removed. FALSE the first column will be kept.
combinedSample	Boolean. If TRUE, importSEQC returns a SingleCellExperiment object containing the combined count matrix, feature annotations and the cell annotations. If FALSE, importSEQC returns a list containing multiple SingleCellExperiment objects. Each SingleCellExperiment contains count matrix, feature annotations and cell annotations for each sample.

Details

importSEQC imports output from seqc. The default sparse_counts_barcode.csv or sparse_counts_genes.csv from seqc output contains two columns. The first column is row index and the second column is cell-barcode or gene symbol. importSEQC will remove first column. Alternatively, user can call cbNotFirstCol or feNotFirstCol as FALSE to keep the first column of these files. When combinedSample is TRUE, importSEQC will combined count matrix with genes detected in at least one sample.

Value

A SingleCellExperiment object containing the combined count matrix, the feature annotations, and the cell annotation.

Examples

```
# Example #1
# The following filtered feature, cell, and matrix files were downloaded from
# https://support.10xgenomics.com/single-cell-gene-expression/datasets/
# 3.0.0/pbmc_1k_v3
# The top 50 hg38 genes are included in this example.
# Only the top 50 cells are included.
```

```
sce <- importSEQC(
    seqcDirs = system.file("extdata/pbmc_1k_50x50", package = "singleCellTK"),
    samples = "pbmc_1k_50x50",
    prefix = "pbmc_1k",
    combinedSample = FALSE)
```

`importSTARsolo` *Construct SCE object from STARsolo outputs*

Description

Read the barcodes, features (genes), and matrices from STARsolo outputs. Import them as one SingleCellExperiment object.

Usage

```
importSTARsolo(
  STARsoloDirs,
  samples,
  STARsoloOuts = "Gene/filtered",
  matrixFileNames = "matrix.mtx",
  featuresFileNames = "features.tsv",
  barcodesFileNames = "barcodes.tsv",
  gzipped = "auto",
  class = c("Matrix", "matrix"),
  delayedArray = FALSE
)
```

Arguments

`STARsoloDirs`	A vector of root directories of STARsolo output files. The paths should be something like this: **/PATH/TO/*prefix*Solo.out**. For example: `./Solo.out`. Each sample should have its own path. Must have the same length as `samples`.
`samples`	A vector of user-defined sample names for the sample to be imported. Must have the same length as `STARsoloDirs`.
`STARsoloOuts`	Character vector. The intermediate paths to filtered or raw cell barcode, feature, and matrix files for each of `samples`. Default `"Gene/filtered"` which works for STAR 2.7.3a. Must have length 1 or the same length as `samples`.
`matrixFileNames`	Filenames for the Market Exchange Format (MEX) sparse matrix file (.mtx file). Must have length 1 or the same length as `samples`.
`featuresFileNames`	Filenames for the feature annotation file. Must have length 1 or the same length as `samples`.
`barcodesFileNames`	Filenames for the cell barcode list file. Must have length 1 or the same length as `samples`.

gzipped	Boolean. TRUE if the STARsolo output files (barcodes.tsv, features.tsv, and matrix.mtx) were gzip compressed. FALSE otherwise. This is FALSE in STAR 2.7.3a. Default "auto" which automatically detects if the files are gzip compressed. Must have length 1 or the same length as samples.
class	Character. The class of the expression matrix stored in the SCE object. Can be one of "Matrix" (as returned by readMM function), or "matrix" (as returned by matrix function). Default "Matrix".
delayedArray	Boolean. Whether to read the expression matrix as DelayedArray object or not. Default FALSE.

Value

A SingleCellExperiment object containing the count matrix, the gene annotation, and the cell annotation.

Examples

```
# Example #1
# FASTQ files were downloaded from
# https://support.10xgenomics.com/single-cell-gene-expression/datasets/3.0.0
# /pbmc_1k_v3
# They were concatenated as follows:
# cat pbmc_1k_v3_S1_L001_R1_001.fastq.gz pbmc_1k_v3_S1_L002_R1_001.fastq.gz >
# pbmc_1k_v3_R1.fastq.gz
# cat pbmc_1k_v3_S1_L001_R2_001.fastq.gz pbmc_1k_v3_S1_L002_R2_001.fastq.gz >
# pbmc_1k_v3_R2.fastq.gz
# The following STARsolo command generates the filtered feature, cell, and
# matrix files
# STAR \
#   --genomeDir ./index \
#   --readFilesIn ./pbmc_1k_v3_R2.fastq.gz \
#                 ./pbmc_1k_v3_R1.fastq.gz \
#   --readFilesCommand zcat \
#   --outSAMtype BAM Unsorted \
#   --outBAMcompression -1 \
#   --soloType CB_UMI_Simple \
#   --soloCBwhitelist ./737K-august-2016.txt \
#   --soloUMIlen 12

# The top 20 genes and the first 20 cells are included in this example.
sce <- importSTARsolo(
  STARsoloDirs = system.file("extdata/STARsolo_PBMC_1k_v3_20x20",
    package = "singleCellTK"),
  samples = "PBMC_1k_v3_20x20")
```

`iterateSimulations` *Returns significance data from a snapshot.*

Description

Returns significance data from a snapshot.

Usage

```
iterateSimulations(
  originalData,
  useAssay = "counts",
  realLabels,
  totalReads,
  cells,
  iterations
)
```

Arguments

originalData	The SingleCellExperiment object storing all assay data from the shiny app.
useAssay	Character. The name of the assay to be used for subsampling.
realLabels	Character. The name of the condition of interest. Must match a name from sample data.
totalReads	Numeric. The total number of reads in the simulated dataset, to be split between all simulated cells.
cells	Numeric. The number of virtual cells to simulate.
iterations	Numeric. How many times should each experimental design be simulated.

Value

A matrix of significance information from a snapshot

Examples

```
data("mouseBrainSubsetSCE")
res <- iterateSimulations(mouseBrainSubsetSCE, realLabels = "level1class",
                          totalReads = 1000, cells = 10, iterations = 2)
```

---

mergeSCEColData *Merging colData from two singleCellExperiment objects*

Description

Merges colData of the singleCellExperiment objects obtained from the same dataset which contain differing colData. (i.e. raw data and filtered data)

Usage

```
mergeSCEColData(inSCE1, inSCE2, id1 = "column_name", id2 = "column_name")
```

Arguments

inSCE1	Input SingleCellExperiment object. The function will output this singleCellExperiment object with a combined colData from inSCE1 and inSCE2.
inSCE2	Input SingleCellExperiment object. colData from this object will be merged with colData from inSCE1 and loaded into inSCE1.
id1	Character vector. Column in colData of inSCE1 that will be used to combine inSCE1 and inSCE2. Default "column_name"
id2	Character vector. Column in colData of inSCE2 that will be used to combine inSCE1 and inSCE2. Default "column_name"

Value

SingleCellExperiment object containing combined colData from both singleCellExperiment for samples in inSCE1.

Examples

```
sce1 <- importCellRanger(
    cellRangerDirs = system.file("extdata/", package = "singleCellTK"),
    sampleDirs = "hgmm_1k_v3_20x20",
    sampleNames = "hgmm1kv3",
    dataType = "filtered")
data(scExample)
sce2 <- sce
sce <- mergeSCEColData(inSCE1 = sce1, inSCE2 = sce2, id1 = "column_name", id2 = "column_name")
```

MitoGenes *List of mitochondrial genes of multiple reference*

Description

A list of gene set that contains mitochondrial genes of multiple reference (hg38, hg19, mm10 and mm9). It contains multiple types of gene identifier: gene symbol, entrez ID, ensemble ID and ensemble transcript ID. It's used for the function 'importMitoGeneSet'.

Usage

```
MitoGenes
```

Format

A list

Examples

```
data("MitoGenes")
```

mouseBrainSubsetSCE *Example Single Cell RNA-Seq data in SingleCellExperiment Object, GSE60361 subset*

Description

A subset of 30 cells from a single cell RNA-Seq experiment from Zeisel, et al. Science 2015. The data was produced from cells from the mouse somatosensory cortex (S1) and hippocampus (CA1). 15 of the cells were identified as oligodendrocytes and 15 of the cell were identified as microglia.

Usage

```
mouseBrainSubsetSCE
```

Format

SingleCellExperiment

Source

DOI: 10.1126/science.aaa1934

Examples

```
data("mouseBrainSubsetSCE")
```

msigdb_table *MSigDB gene get Cctegory table*

Description

A table of gene set categories that can be download from MSigDB. The categories and descriptions can be found here: https://www.gsea-msigdb.org/gsea/msigdb/collections.jsp. The IDs in the first column can be used to retrieve the gene sets for these categories using the importGeneSetsFromMSigDB function.

Usage

```
msigdb_table
```

Format

A data.frame.

Examples

```
data("msigdb_table")
```

plotBarcodeRankDropsResults *Plots for runEmptyDrops outputs.*

Description

A wrapper function which visualizes outputs from the runEmptyDrops function stored in the colData slot of the SingleCellExperiment object via plots.

Usage

```
plotBarcodeRankDropsResults(
  inSCE,
  sample = NULL,
  defaultTheme = TRUE,
  dotSize = 0.5,
  titleSize = 18,
  axisLabelSize = 18,
  axisSize = 15,
  legendSize = 15
)
```

Arguments

inSCE	Input SingleCellExperiment object with saved dimension reduction components or a variable with saved results from runBarcodeRankDrops. Required.
sample	Character vector. Indicates which sample each cell belongs to. Default NULL.
defaultTheme	Removes grid in plot and sets axis title size to 10 when TRUE. Default TRUE.
dotSize	Size of dots. Default 0.5.
titleSize	Size of title of plot. Default 18.
axisLabelSize	Size of x/y-axis labels. Default 18.
axisSize	Size of x/y-axis ticks. Default 15.
legendSize	size of legend. Default 15.

Value

list of .ggplot objects

Examples

```
data(scExample, package="singleCellTK")
sce <- runBarcodeRankDrops(inSCE=sce)
plotBarcodeRankDropsResults(inSCE=sce)
```

`plotBarcodeRankScatter` *Plots for runBarcodeRankDrops outputs.*

Description

A plotting function which visualizes outputs from the runBarcodeRankDrops function stored in the colData slot of the SingleCellExperiment object via scatterplot.

Usage

```
plotBarcodeRankScatter(
  inSCE,
  sample = NULL,
  defaultTheme = TRUE,
  dotSize = 0.5,
  title = NULL,
  titleSize = 18,
  xlab = NULL,
  ylab = NULL,
  axisSize = 12,
  axisLabelSize = 15,
  legendSize = 10,
  combinePlot = "none",
  sampleRelHeights = 1,
  sampleRelWidths = 1
)
```

Arguments

`inSCE`	Input SingleCellExperiment object with saved dimension reduction components or a variable with saved results from runBarcodeRankDrops. Required.
`sample`	Character vector. Indicates which sample each cell belongs to. Default NULL.
`defaultTheme`	Removes grid in plot and sets axis title size to 10 when TRUE. Default TRUE.
`dotSize`	Size of dots. Default 0.5.
`title`	Title of plot. Default NULL.
`titleSize`	Size of title of plot. Default 18.
`xlab`	Character vector. Label for x-axis. Default NULL.
`ylab`	Character vector. Label for y-axis. Default NULL.
`axisSize`	Size of x/y-axis ticks. Default 12.
`axisLabelSize`	Size of x/y-axis labels. Default 15.
`legendSize`	size of legend. Default 10.
`combinePlot`	Boolean. If multiple plots are generated (multiple samples, etc.), will combined plots using ‘cowplot::plot_grid‘.

sampleRelHeights
: If there are multiple samples and combining by "all", the relative heights for each plot.

sampleRelWidths
: If there are multiple samples and combining by "all", the relative widths for each plot. Default TRUE.

Value

a ggplot object of the scatter plot.

Examples

```
data(scExample, package="singleCellTK")
sce <- runBarcodeRankDrops(inSCE=sce)
plotBarcodeRankScatter(inSCE=sce)
```

---

plotBatchCorrCompare *Plot comparison of batch corrected result against original assay*

Description

Plot comparison of batch corrected result against original assay

Usage

```
plotBatchCorrCompare(
  inSCE,
  corrMat,
  batch = NULL,
  condition = NULL,
  origAssay = NULL,
  origLogged = NULL,
  method = NULL,
  matType = NULL
)
```

Arguments

inSCE	SingleCellExperiment inherited object.
corrMat	A single character indicating the name of the corrected matrix.
batch	A single character. The name of batch annotation column in `colData(inSCE)`.
condition	A single character. The name of an additional covariate annotation column in `colData(inSCE)`.
origAssay	A single character indicating what the original assay used for batch correction is.

origLogged	Logical scalar indicating whether origAssay is log-normalized.
method	A single character indicating the name of the batch correction method. Only used for the titles of plots.
matType	A single character indicating the type of the batch correction result matrix, choose from "assay", "altExp", "reducedDim".

Details

Four plots will be combined. Two of them are violin/box-plots for percent variance explained by the batch variation, and optionally the covariate, for original and corrected. The other two are UMAPs of the original assay and the correction result matrix. If SCTK batch correction methods are performed in advance, this function will automatically detect necessary input. Otherwise, users can also customize the input. Future improvement might include solution to reduce redundant UMAP calculation.

Value

An object of class "gtable", combining four ggplots.

Author(s)

Yichen Wang

Examples

```
sceBatches <- scaterlogNormCounts(sceBatches, "logcounts")
sceBatches <- runLimmaBC(sceBatches)
plotBatchCorrCompare(sceBatches, "LIMMA", condition = "cell_type")
```

---

plotBatchVariance *Plot the percent of the variation that is explained by batch and condition in the data*

Description

Visualize the percent variation in the data that is explained by batch and condition, individually, and that explained by combining both annotations. Plotting only the variation explained by batch is supported but not recommended, because this can be confounded by potential condition.

Usage

```
plotBatchVariance(
  inSCE,
  useAssay = NULL,
  useReddim = NULL,
  useAltExp = NULL,
  batch = "batch",
  condition = NULL,
```

```
  title = NULL
)
```

Arguments

inSCE	SingleCellExperiment inherited object.
useAssay	A single character. The name of the assay that stores the value to plot. For useReddim and useAltExp also. Default NULL.
useReddim	A single character. The name of the dimension reduced matrix that stores the value to plot. Default NULL.
useAltExp	A single character. The name of the alternative experiment that stores an assay of the value to plot. Default NULL.
batch	A single character. The name of batch annotation column in colData(inSCE). Default "batch".
condition	A single character. The name of an additional condition annotation column in colData(inSCE). Default NULL.
title	A single character. The title text on the top. Default NULL.

Details

When condition and batch both are causing some variation, if the difference between full variation and condition variation is close to batch variation, this might imply that batches are causing some effect; if the difference is much less than batch variation, then the batches are likely to be confounded by the conditions.

Value

A ggplot object of a boxplot of variation explained by batch, condition, and batch+condition.

Examples

```
data('sceBatches', package = 'singleCellTK')
plotBatchVariance(sceBatches,
                  useAssay="counts",
                  batch="batch",
                  condition = "cell_type")
```

---

plotBcdsResults *Plots for runBcds outputs.*

Description

A wrapper function which visualizes outputs from the runBcds function stored in the colData slot of the SingleCellExperiment object via various plots.

Usage

```
plotBcdsResults(
  inSCE,
  sample = NULL,
  shape = NULL,
  groupBy = NULL,
  combinePlot = "all",
  violin = TRUE,
  boxplot = FALSE,
  dots = TRUE,
  reducedDimName = "UMAP",
  xlab = NULL,
  ylab = NULL,
  dim1 = NULL,
  dim2 = NULL,
  bin = NULL,
  binLabel = NULL,
  defaultTheme = TRUE,
  dotSize = 0.5,
  summary = "median",
  summaryTextSize = 3,
  transparency = 1,
  baseSize = 15,
  titleSize = NULL,
  axisLabelSize = NULL,
  axisSize = NULL,
  legendSize = NULL,
  legendTitleSize = NULL,
  relHeights = 1,
  relWidths = c(1, 1, 1),
  plotNCols = NULL,
  plotNRows = NULL,
  labelSamples = TRUE,
  samplePerColumn = TRUE,
  sampleRelHeights = 1,
  sampleRelWidths = 1
)
```

Arguments

inSCE	Input SingleCellExperiment object with saved dimension reduction components or a variable with saved results from runBcds. Required.
sample	Character vector. Indicates which sample each cell belongs to. Default NULL.
shape	If provided, add shapes based on the value.
groupBy	Groupings for each numeric value. A user may input a vector equal length to the number of the samples in the SingleCellExperiment object, or can be retrieved from the colData slot. Default NULL.

combinePlot	Must be either "all", "sample", or "none". "all" will combine all plots into a single .ggplot object, while "sample" will output a list of plots separated by sample. Default "all".
violin	Boolean. If TRUE, will plot the violin plot. Default TRUE.
boxplot	Boolean. If TRUE, will plot boxplots for each violin plot. Default TRUE.
dots	Boolean. If TRUE, will plot dots for each violin plot. Default TRUE.
reducedDimName	Saved dimension reduction name in the SingleCellExperiment object. Required.
xlab	Character vector. Label for x-axis. Default NULL.
ylab	Character vector. Label for y-axis. Default NULL.
dim1	1st dimension to be used for plotting. Can either be a string which specifies the name of the dimension to be plotted from reducedDims, or a numeric value which specifies the index of the dimension to be plotted. Default is NULL.
dim2	2nd dimension to be used for plotting. Can either be a string which specifies the name of the dimension to be plotted from reducedDims, or a numeric value which specifies the index of the dimension to be plotted. Default is NULL.
bin	Numeric vector. If single value, will divide the numeric values into the 'bin' groups. If more than one value, will bin numeric values using values as a cut point.
binLabel	Character vector. Labels for the bins created by the 'bin' parameter. Default NULL.
defaultTheme	Removes grid in plot and sets axis title size to 10 when TRUE. Default TRUE.
dotSize	Size of dots. Default 0.5.
summary	Adds a summary statistic, as well as a crossbar to the violin plot. Options are "mean" or "median". Default NULL.
summaryTextSize	The text size of the summary statistic displayed above the violin plot. Default 3.
transparency	Transparency of the dots, values will be 0-1. Default 1.
baseSize	The base font size for all text. Default 15. Can be overwritten by titleSize, axisSize, and axisLabelSize, legendSize, legendTitleSize.
titleSize	Size of title of plot. Default NULL.
axisLabelSize	Size of x/y-axis labels. Default NULL.
axisSize	Size of x/y-axis ticks. Default NULL.
legendSize	size of legend. Default NULL.
legendTitleSize	size of legend title. Default NULL.
relHeights	Relative heights of plots when combine is set.
relWidths	Relative widths of plots when combine is set.
plotNCols	Number of columns when plots are combined in a grid.
plotNRows	Number of rows when plots are combined in a grid.
labelSamples	Will label sample name in title of plot if TRUE. Default TRUE.

samplePerColumn
: If TRUE, when there are multiple samples and combining by "all", the output .ggplot will have plots from each sample on a single column. Default TRUE.

sampleRelHeights
: If there are multiple samples and combining by "all", the relative heights for each plot.

sampleRelWidths
: If there are multiple samples and combining by "all", the relative widths for each plot.

Value

list of .ggplot objects

Examples

```
data(scExample, package="singleCellTK")
sce <- subsetSCECols(sce, colData = "type != 'EmptyDroplet'")
sce <- getUMAP(inSCE=sce, useAssay="counts", reducedDimName="UMAP")
sce <- runBcds(sce)
plotBcdsResults(inSCE=sce, reducedDimName="UMAP")
```

---

plotClusterAbundance *Plot the differential Abundance*

Description

Plot the differential Abundance

Usage

```
plotClusterAbundance(inSCE, cluster, variable)
```

Arguments

inSCE	A SingleCellExperiment object.
cluster	A single character, specifying the name to store the cluster label in colData.
variable	A single character, specifying the name to store the phenotype labels in colData.

Details

This function will visualize the differential abundance in two given variables, by making bar plots that presents the cell counting and fraction in different cases.

Value

A list with 4 ggplot objects.

Examples

```
data("mouseBrainSubsetSCE", package = "singleCellTK")
plotClusterAbundance(inSCE = mouseBrainSubsetSCE,
                     cluster = "tissue",
                     variable = "level1class")
```

`plotCxdsResults` *Plots for runCxds outputs.*

Description

A wrapper function which visualizes outputs from the runCxds function stored in the colData slot of the SingleCellExperiment object via various plots.

Usage

```
plotCxdsResults(
  inSCE,
  sample = NULL,
  shape = NULL,
  groupBy = NULL,
  combinePlot = "all",
  violin = TRUE,
  boxplot = FALSE,
  dots = TRUE,
  reducedDimName = "UMAP",
  xlab = NULL,
  ylab = NULL,
  dim1 = NULL,
  dim2 = NULL,
  bin = NULL,
  binLabel = NULL,
  defaultTheme = TRUE,
  dotSize = 0.5,
  summary = "median",
  summaryTextSize = 3,
  transparency = 1,
  baseSize = 15,
  titleSize = NULL,
  axisLabelSize = NULL,
  axisSize = NULL,
  legendSize = NULL,
  legendTitleSize = NULL,
  relHeights = 1,
  relWidths = c(1, 1, 1),
  plotNCols = NULL,
  plotNRows = NULL,
```

```
  labelSamples = TRUE,
  samplePerColumn = TRUE,
  sampleRelHeights = 1,
  sampleRelWidths = 1
)
```

Arguments

inSCE	Input SingleCellExperiment object with saved dimension reduction components or a variable with saved results from runCxds.
sample	Character vector. Indicates which sample each cell belongs to. Default NULL.
shape	If provided, add shapes based on the value.
groupBy	Groupings for each numeric value. A user may input a vector equal length to the number of the samples in the SingleCellExperiment object, or can be retrieved from the colData slot. Default NULL.
combinePlot	Must be either "all", "sample", or "none". "all" will combine all plots into a single .ggplot object, while "sample" will output a list of plots separated by sample. Default "all".
violin	Boolean. If TRUE, will plot the violin plot. Default TRUE.
boxplot	Boolean. If TRUE, will plot boxplots for each violin plot. Default TRUE.
dots	Boolean. If TRUE, will plot dots for each violin plot. Default TRUE.
reducedDimName	Saved dimension reduction name in the SingleCellExperiment object. Required.
xlab	Character vector. Label for x-axis. Default NULL.
ylab	Character vector. Label for y-axis. Default NULL.
dim1	1st dimension to be used for plotting. Can either be a string which specifies the name of the dimension to be plotted from reducedDims, or a numeric value which specifies the index of the dimension to be plotted. Default is NULL.
dim2	2nd dimension to be used for plotting. Can either be a string which specifies the name of the dimension to be plotted from reducedDims, or a numeric value which specifies the index of the dimension to be plotted. Default is NULL.
bin	Numeric vector. If single value, will divide the numeric values into the ‘bin‘ groups. If more than one value, will bin numeric values using values as a cut point.
binLabel	Character vector. Labels for the bins created by the ‘bin‘ parameter. Default NULL.
defaultTheme	Removes grid in plot and sets axis title size to 10 when TRUE. Default TRUE.
dotSize	Size of dots. Default 0.5.
summary	Adds a summary statistic, as well as a crossbar to the violin plot. Options are "mean" or "median". Default NULL.
summaryTextSize	The text size of the summary statistic displayed above the violin plot. Default 3.
transparency	Transparency of the dots, values will be 0-1. Default 1.

baseSize	The base font size for all text. Default 12. Can be overwritten by titleSize, axisSize, and axisLabelSize, legendSize, legendTitleSize.
titleSize	Size of title of plot. Default NULL.
axisLabelSize	Size of x/y-axis labels. Default NULL.
axisSize	Size of x/y-axis ticks. Default NULL.
legendSize	size of legend. Default NULL.
legendTitleSize	size of legend title. Default NULL.
relHeights	Relative heights of plots when combine is set.
relWidths	Relative widths of plots when combine is set.
plotNCols	Number of columns when plots are combined in a grid.
plotNRows	Number of rows when plots are combined in a grid.
labelSamples	Will label sample name in title of plot if TRUE. Default TRUE.
samplePerColumn	If TRUE, when there are multiple samples and combining by "all", the output .ggplot will have plots from each sample on a single column. Default TRUE.
sampleRelHeights	If there are multiple samples and combining by "all", the relative heights for each plot.
sampleRelWidths	If there are multiple samples and combining by "all", the relative widths for each plot.

Value

list of .ggplot objects

Examples

```
data(scExample, package="singleCellTK")
sce <- subsetSCECols(sce, colData = "type != 'EmptyDroplet'")
sce <- getUMAP(inSCE=sce, useAssay="counts", reducedDimName="UMAP")
sce <- runCxds(sce)
plotCxdsResults(inSCE=sce, reducedDimName="UMAP")
```

---

plotDecontXResults *Plots for runDecontX outputs.*

Description

A wrapper function which visualizes outputs from the runDecontX function stored in the colData slot of the SingleCellExperiment object via various plots.

Usage

```
plotDecontXResults(
  inSCE,
  sample = NULL,
  shape = NULL,
  groupBy = NULL,
  combinePlot = "all",
  violin = TRUE,
  boxplot = FALSE,
  dots = TRUE,
  reducedDimName = "UMAP",
  xlab = NULL,
  ylab = NULL,
  dim1 = NULL,
  dim2 = NULL,
  bin = NULL,
  binLabel = NULL,
  defaultTheme = TRUE,
  dotSize = 0.5,
  summary = "median",
  summaryTextSize = 3,
  transparency = 1,
  baseSize = 15,
  titleSize = NULL,
  axisLabelSize = NULL,
  axisSize = NULL,
  legendSize = NULL,
  legendTitleSize = NULL,
  relHeights = 1,
  relWidths = c(1, 1, 1),
  plotNCols = NULL,
  plotNRows = NULL,
  labelSamples = TRUE,
  labelClusters = TRUE,
  clusterLabelSize = 3.5,
  samplePerColumn = TRUE,
  sampleRelHeights = 1,
  sampleRelWidths = 1
)
```

Arguments

inSCE	Input SingleCellExperiment object with saved dimension reduction components or a variable with saved results from runDecontX. Required.
sample	Character vector. Indicates which sample each cell belongs to. Default NULL.
shape	If provided, add shapes based on the value.
groupBy	Groupings for each numeric value. A user may input a vector equal length to the

	number of the samples in the SingleCellExperiment object, or can be retrieved from the colData slot. Default NULL.
combinePlot	Must be either "all", "sample", or "none". "all" will combine all plots into a single .ggplot object, while "sample" will output a list of plots separated by sample. Default "all".
violin	Boolean. If TRUE, will plot the violin plot. Default TRUE.
boxplot	Boolean. If TRUE, will plot boxplots for each violin plot. Default TRUE.
dots	Boolean. If TRUE, will plot dots for each violin plot. Default TRUE.
reducedDimName	Saved dimension reduction name in the SingleCellExperiment object. Required. Default = "UMAP"
xlab	Character vector. Label for x-axis. Default NULL.
ylab	Character vector. Label for y-axis. Default NULL.
dim1	1st dimension to be used for plotting. Can either be a string which specifies the name of the dimension to be plotted from reducedDims, or a numeric value which specifies the index of the dimension to be plotted. Default is NULL.
dim2	2nd dimension to be used for plotting. Can either be a string which specifies the name of the dimension to be plotted from reducedDims, or a numeric value which specifies the index of the dimension to be plotted. Default is NULL.
bin	Numeric vector. If single value, will divide the numeric values into the 'bin' groups. If more than one value, will bin numeric values using values as a cut point.
binLabel	Character vector. Labels for the bins created by the 'bin' parameter. Default NULL.
defaultTheme	Removes grid in plot and sets axis title size to 10 when TRUE. Default TRUE.
dotSize	Size of dots. Default 0.5.
summary	Adds a summary statistic, as well as a crossbar to the violin plot. Options are "mean" or "median". Default NULL.
summaryTextSize	The text size of the summary statistic displayed above the violin plot. Default 3.
transparency	Transparency of the dots, values will be 0-1. Default 1.
baseSize	The base font size for all text. Default 12. Can be overwritten by titleSize, axisSize, and axisLabelSize, legendSize, legendTitleSize.
titleSize	Size of title of plot. Default NULL.
axisLabelSize	Size of x/y-axis labels. Default NULL.
axisSize	Size of x/y-axis ticks. Default NULL.
legendSize	size of legend. Default NULL.
legendTitleSize	size of legend title. Default NULL.
relHeights	Relative heights of plots when combine is set.
relWidths	Relative widths of plots when combine is set.
plotNCols	Number of columns when plots are combined in a grid.

plotNRows Number of rows when plots are combined in a grid.

labelSamples Will label sample name in title of plot if TRUE. Default TRUE.

labelClusters Logical. Whether the cluster labels are plotted. Default FALSE.

clusterLabelSize
Numeric. Determines the size of cluster label when ‘labelClusters‘ is set to TRUE. Default 3.5.

samplePerColumn
If TRUE, when there are multiple samples and combining by "all", the output .ggplot will have plots from each sample on a single column. Default TRUE.

sampleRelHeights
If there are multiple samples and combining by "all", the relative heights for each plot.

sampleRelWidths
If there are multiple samples and combining by "all", the relative widths for each plot.

Value

list of .ggplot objects

Examples

```
data(scExample, package="singleCellTK")
sce <- subsetSCECols(sce, colData = "type != 'EmptyDroplet'")
sce <- runDecontX(sce)
plotDecontXResults(inSCE=sce, reducedDimName="decontX_UMAP")
```

---

plotDEGHeatmap *Heatmap visualization of DEG result*

Description

A differential expression analysis function has to be run in advance so that information is stored in the metadata of the input SCE object. This function wraps plotSCEHeatmap. A feature annotation basing on the log2FC level called "regulation" will be automatically added. A cell annotation basing on the condition selection while running the analysis called "condition", and the annotations used from colData(inSCE) while setting the condition and covariates will also be added.

Usage

```
plotDEGHeatmap(
  inSCE,
  useResult,
  doLog = FALSE,
  onlyPos = FALSE,
  log2fcThreshold = 0.25,
```

```
  fdrThreshold = 0.05,
  useAssay = NULL,
  featureAnnotations = NULL,
  cellAnnotations = NULL,
  featureAnnotationColor = NULL,
  cellAnnotationColor = NULL,
  rowDataName = NULL,
  colDataName = NULL,
  colSplitBy = "condition",
  rowSplitBy = "regulation",
  title = paste0("DE Analysis: ", useResult),
  ...
)
```

Arguments

inSCE	SingleCellExperiment inherited object. runMAST() has to be run in advance.
useResult	character. A string specifying the analysisName used when running a differential expression analysis function.
doLog	Logical scalar. Whether to do log(assay + 1) transformation on the assay used for the analysis. Default FALSE.
onlyPos	logical. Whether to only plot DEG with positive log2_FC value. Default FALSE.
log2fcThreshold	numeric. Only plot DEGs with the absolute values of log2FC larger than this value. Default 0.25.
fdrThreshold	numeric. Only plot DEGs with FDR value smaller than this value. Default 0.05.
useAssay	character. A string specifying an assay of expression value to plot. By default the assay used for runMAST() will be used. Default NULL.
featureAnnotations	data.frame, with rownames containing all the features going to be plotted. Character columns should be factors. Default NULL.
cellAnnotations	data.frame, with rownames containing all the cells going to be plotted. Character columns should be factors. Default NULL.
featureAnnotationColor	A named list. Customized color settings for feature labeling. Should match the entries in the featureAnnotations or rowDataName. For each entry, there should be a list/vector of colors named with categories. Default NULL.
cellAnnotationColor	A named list. Customized color settings for cell labeling. Should match the entries in the cellAnnotations or colDataName. For each entry, there should be a list/vector of colors named with categories. Default NULL.
rowDataName	character. The column name(s) in rowData that need to be added to the annotation. Default NULL.
colDataName	character. The column name(s) in colData that need to be added to the annotation. Default NULL.

colSplitBy	character. Do semi-heatmap based on the grouping of this(these) annotation(s). Should exist in either colDataName or names(cellAnnotations). Default "condition".
rowSplitBy	character. Do semi-heatmap based on the grouping of this(these) annotation(s). Should exist in either rowDataName or names(featureAnnotations). Default "regulation".
title	character. Main title of the heatmap. Default "DE Analysis: <useResult>".
...	Other arguments passed to plotSCEHeatmap

Value

A ComplexHeatmap::Heatmap object

Author(s)

Yichen Wang

Examples

```
data("sceBatches")
logcounts(sceBatches) <- log(counts(sceBatches) + 1)
sce.w <- subsetSCECols(sceBatches, colData = "batch == 'w'")
sce.w <- runWilcox(sce.w, class = "cell_type", classGroup1 = "alpha",
                   groupName1 = "w.alpha", groupName2 = "w.beta",
                   analysisName = "w.aVSb")
plotDEGHeatmap(sce.w, "w.aVSb")
```

---

plotDEGRegression *plot the linear regression to show visualize the expression the of DEGs identified by differential expression analysis*

Description

plot the linear regression to show visualize the expression the of DEGs identified by differential expression analysis

Usage

```
plotDEGRegression(
  inSCE,
  useResult,
  threshP = FALSE,
  labelBy = NULL,
  nrow = 6,
  ncol = 6,
  defaultTheme = TRUE,
  isLogged = TRUE,
  check_sanity = TRUE
)
```

Arguments

inSCE	SingleCellExperiment inherited object. runMAST() has to be run in advance.
useResult	character. A string specifying the analysisName used when running a differential expression analysis function.
threshP	logical. Whether to plot threshold values from adaptive thresholding, instead of using the assay used by runMAST(). Default FALSE.
labelBy	A single character for a column of rowData(inSCE) as where to search for the labeling text. Default NULL.
nrow	Integer. Number of rows in the plot grid. Default 6.
ncol	Integer. Number of columns in the plot grid. Default 6.
defaultTheme	Logical scalar. Whether to use default SCTK theme in ggplot. Default TRUE.
isLogged	Logical scalar. Whether the assay used for the analysis is logged. If not, will do a log(assay + 1) transformation. Default TRUE.
check_sanity	Logical scalar. Whether to perform MAST's sanity check to see if the counts are logged. Default TRUE

Value

A ggplot object of linear regression

Examples

```
data("sceBatches")
logcounts(sceBatches) <- log(counts(sceBatches) + 1)
sce.w <- subsetSCECols(sceBatches, colData = "batch == 'w'")
sce.w <- runWilcox(sce.w, class = "cell_type", classGroup1 = "alpha",
                   groupName1 = "w.alpha", groupName2 = "w.beta",
                   analysisName = "w.aVSb")
plotDEGRegression(sce.w, "w.aVSb")
```

---

plotDEGViolin *plot the violin plot to show visualize the expression distribution of DEGs identified by differential expression analysis*

Description

plot the violin plot to show visualize the expression distribution of DEGs identified by differential expression analysis

Usage

```
plotDEGViolin(
  inSCE,
  useResult,
  threshP = FALSE,
  labelBy = NULL,
  nrow = 6,
  ncol = 6,
  defaultTheme = TRUE,
  isLogged = TRUE,
  check_sanity = TRUE
)
```

Arguments

inSCE	SingleCellExperiment inherited object. runMAST() has to be run in advance.
useResult	character. A string specifying the analysisName used when running a differential expression analysis function.
threshP	logical. Whether to plot threshold values from adaptive thresholding, instead of using the assay used by runMAST(). Default FALSE.
labelBy	A single character for a column of rowData(inSCE) as where to search for the labeling text. Default NULL.
nrow	Integer. Number of rows in the plot grid. Default 6.
ncol	Integer. Number of columns in the plot grid. Default 6.
defaultTheme	Logical scalar. Whether to use default SCTK theme in ggplot. Default TRUE.
isLogged	Logical scalar. Whether the assay used for the analysis is logged. If not, will do a log(assay + 1) transformation. Default TRUE.
check_sanity	Logical scalar. Whether to perform MAST's sanity check to see if the counts are logged. Default TRUE

Value

A ggplot object of violin plot

Examples

```
data("sceBatches")
logcounts(sceBatches) <- log(counts(sceBatches) + 1)
sce.w <- subsetSCECols(sceBatches, colData = "batch == 'w'")
sce.w <- runWilcox(sce.w, class = "cell_type", classGroup1 = "alpha",
                   groupName1 = "w.alpha", groupName2 = "w.beta",
                   analysisName = "w.aVSb")
plotDEGViolin(sce.w, "w.aVSb")
```

plotDimRed *Plot dimensionality reduction from computed metrics including PCA, ICA, tSNE and UMAP*

Description

Plot dimensionality reduction from computed metrics including PCA, ICA, tSNE and UMAP

Usage

```
plotDimRed(
  inSCE,
  useReduction,
  showLegend = FALSE,
  xDim = 1,
  yDim = 2,
  xAxisLabel = NULL,
  yAxisLabel = NULL
)
```

Arguments

inSCE	Input SCE object
useReduction	Reduction to plot
showLegend	If legends should be plotted or not
xDim	Numeric value indicating the dimension to use for X-axis. Default is 1 (refers to PC1).
yDim	Numeric value indicating the dimension to use for Y-axis. Default is 2 (refers to PC2).
xAxisLabel	Specify the label for x-axis. Default is NULL which will specify the label as 'x'.
yAxisLabel	Specify the label for y-axis. Default is NULL which will specify the label as 'y'.

Value

plot object

Examples

```
data("mouseBrainSubsetSCE", package = "singleCellTK")
plotDimRed(mouseBrainSubsetSCE, "PCA_logcounts")
```

plotDoubletFinderResults
Plots for runDoubletFinder outputs.

Description

A wrapper function which visualizes outputs from the runDoubletFinder function stored in the colData slot of the SingleCellExperiment object via various plots.

Usage

```
plotDoubletFinderResults(
  inSCE,
  sample = NULL,
  shape = NULL,
  groupBy = NULL,
  combinePlot = "all",
  violin = TRUE,
  boxplot = FALSE,
  dots = TRUE,
  reducedDimName = "UMAP",
  xlab = NULL,
  ylab = NULL,
  dim1 = NULL,
  dim2 = NULL,
  bin = NULL,
  binLabel = NULL,
  defaultTheme = TRUE,
  dotSize = 0.5,
  summary = "median",
  summaryTextSize = 3,
  transparency = 1,
  baseSize = 15,
  titleSize = NULL,
  axisLabelSize = NULL,
  axisSize = NULL,
  legendSize = NULL,
  legendTitleSize = NULL,
  relHeights = 1,
  relWidths = c(1, 1, 1),
  plotNCols = NULL,
  plotNRows = NULL,
  labelSamples = TRUE,
  samplePerColumn = TRUE,
  sampleRelHeights = 1,
  sampleRelWidths = 1
)
```

Arguments

inSCE	Input SingleCellExperiment object with saved dimension reduction components or a variable with saved results from runDoubletFinder. Required.
sample	Character vector. Indicates which sample each cell belongs to. Default NULL.
shape	If provided, add shapes based on the value.
groupBy	Groupings for each numeric value. A user may input a vector equal length to the number of the samples in the SingleCellExperiment object, or can be retrieved from the colData slot. Default NULL.
combinePlot	Must be either "all", "sample", or "none". "all" will combine all plots into a single .ggplot object, while "sample" will output a list of plots separated by sample. Default "all".
violin	Boolean. If TRUE, will plot the violin plot. Default TRUE.
boxplot	Boolean. If TRUE, will plot boxplots for each violin plot. Default TRUE.
dots	Boolean. If TRUE, will plot dots for each violin plot. Default TRUE.
reducedDimName	Saved dimension reduction name in the SingleCellExperiment object. Required.
xlab	Character vector. Label for x-axis. Default NULL.
ylab	Character vector. Label for y-axis. Default NULL.
dim1	1st dimension to be used for plotting. Can either be a string which specifies the name of the dimension to be plotted from reducedDims, or a numeric value which specifies the index of the dimension to be plotted. Default is NULL.
dim2	2nd dimension to be used for plotting. Can either be a string which specifies the name of the dimension to be plotted from reducedDims, or a numeric value which specifies the index of the dimension to be plotted. Default is NULL.
bin	Numeric vector. If single value, will divide the numeric values into the 'bin' groups. If more than one value, will bin numeric values using values as a cut point.
binLabel	Character vector. Labels for the bins created by the 'bin' parameter. Default NULL.
defaultTheme	Removes grid in plot and sets axis title size to 10 when TRUE. Default TRUE.
dotSize	Size of dots. Default 0.5.
summary	Adds a summary statistic, as well as a crossbar to the violin plot. Options are "mean" or "median". Default NULL.
summaryTextSize	The text size of the summary statistic displayed above the violin plot. Default 3.
transparency	Transparency of the dots, values will be 0-1. Default 1.
baseSize	The base font size for all text. Default 12. Can be overwritten by titleSize, axisSize, and axisLabelSize, legendSize, legendTitleSize.
titleSize	Size of title of plot. Default NULL.
axisLabelSize	Size of x/y-axis labels. Default NULL.
axisSize	Size of x/y-axis ticks. Default NULL.
legendSize	size of legend. Default NULL.

legendTitleSize
: size of legend title. Default NULL.

relHeights
: Relative heights of plots when combine is set.

relWidths
: Relative widths of plots when combine is set.

plotNCols
: Number of columns when plots are combined in a grid.

plotNRows
: Number of rows when plots are combined in a grid.

labelSamples
: Will label sample name in title of plot if TRUE. Default TRUE.

samplePerColumn
: If TRUE, when there are multiple samples and combining by "all", the output .ggplot will have plots from each sample on a single column. Default TRUE.

sampleRelHeights
: If there are multiple samples and combining by "all", the relative heights for each plot.

sampleRelWidths
: If there are multiple samples and combining by "all", the relative widths for each plot.

Value

list of .ggplot objects

Examples

```
data(scExample, package="singleCellTK")
sce <- subsetSCECols(sce, colData = "type != 'EmptyDroplet'")
sce <- getUMAP(inSCE=sce, useAssay="counts", reducedDimName="UMAP")
sce <- runDoubletFinder(sce)
plotDoubletFinderResults(inSCE=sce, reducedDimName="UMAP")
```

---

plotEmptyDropsResults *Plots for runEmptyDrops outputs.*

Description

A wrapper function which visualizes outputs from the runEmptyDrops function stored in the colData slot of the SingleCellExperiment object via plots.

Usage

```
plotEmptyDropsResults(
  inSCE,
  sample = NULL,
  combinePlot = "all",
  fdrCutoff = 0.01,
  defaultTheme = TRUE,
  dotSize = 0.5,
```

```
  titleSize = 18,
  axisLabelSize = 18,
  axisSize = 15,
  legendSize = 15,
  legendTitleSize = 16,
  relHeights = 1,
  relWidths = 1,
  samplePerColumn = TRUE,
  sampleRelHeights = 1,
  sampleRelWidths = 1
)
```

Arguments

inSCE	Input SingleCellExperiment object with saved dimension reduction components or a variable with saved results from runScrublet. Required.
sample	Character vector. Indicates which sample each cell belongs to. Default NULL.
combinePlot	Must be either "all", "sample", or "none". "all" will combine all plots into a single .ggplot object, while "sample" will output a list of plots separated by sample. Default "all".
fdrCutoff	Numeric. Thresholds barcodes based on the FDR values from runEmptyDrops as "Empty Droplet" or "Putative Cell". Default 0.01.
defaultTheme	Removes grid in plot and sets axis title size to 10 when TRUE. Default TRUE.
dotSize	Size of dots. Default 0.5.
titleSize	Size of title of plot. Default 18.
axisLabelSize	Size of x/y-axis labels. Default 18.
axisSize	Size of x/y-axis ticks. Default 15.
legendSize	size of legend. Default 15.
legendTitleSize	size of legend title. Default 16.
relHeights	Relative heights of plots when combine is set.
relWidths	Relative widths of plots when combine is set.
samplePerColumn	If TRUE, when there are multiple samples and combining by "all", the output .ggplot will have plots from each sample on a single column. Default TRUE.
sampleRelHeights	If there are multiple samples and combining by "all", the relative heights for each plot.
sampleRelWidths	If there are multiple samples and combining by "all", the relative widths for each plot.

Value

list of .ggplot objects

Examples

```
data(scExample, package="singleCellTK")
sce <- runEmptyDrops(inSCE=sce)
plotEmptyDropsResults(inSCE=sce)
```

`plotEmptyDropsScatter` *Plots for runEmptyDrops outputs.*

Description

A plotting function which visualizes outputs from the runEmptyDrops function stored in the colData slot of the SingleCellExperiment object via scatterplot.

Usage

```
plotEmptyDropsScatter(
  inSCE,
  sample = NULL,
  fdrCutoff = 0.01,
  defaultTheme = TRUE,
  dotSize = 0.5,
  title = NULL,
  titleSize = 18,
  xlab = NULL,
  ylab = NULL,
  axisSize = 12,
  axisLabelSize = 15,
  legendTitle = NULL,
  legendTitleSize = 12,
  legendSize = 10,
  combinePlot = "none",
  relHeights = 1,
  relWidths = 1,
  samplePerColumn = TRUE,
  sampleRelHeights = 1,
  sampleRelWidths = 1
)
```

Arguments

`inSCE`	Input SingleCellExperiment object with saved dimension reduction components or a variable with saved results from runEmptyDrops. Required.
`sample`	Character vector. Indicates which sample each cell belongs to. Default NULL.
`fdrCutoff`	Numeric. Thresholds barcodes based on the FDR values from runEmptyDrops as "Empty Droplet" or "Putative Cell". Default 0.01.
`defaultTheme`	Removes grid in plot and sets axis title size to 10 when TRUE. Default TRUE.

dotSize	Size of dots. Default 0.5.
title	Title of plot. Default NULL.
titleSize	Size of title of plot. Default 18.
xlab	Character vector. Label for x-axis. Default NULL.
ylab	Character vector. Label for y-axis. Default NULL.
axisSize	Size of x/y-axis ticks. Default 12.
axisLabelSize	Size of x/y-axis labels. Default 15.
legendTitle	Title of legend. Default NULL.
legendTitleSize	size of legend title. Default 12.
legendSize	size of legend. Default 10.
combinePlot	Boolean. If multiple plots are generated (multiple samples, etc.), will combined plots using ‘cowplot::plot_grid‘. Default TRUE.
relHeights	Relative heights of plots when combine is set.
relWidths	Relative widths of plots when combine is set.
samplePerColumn	If TRUE, when there are multiple samples and combining by "all", the output .ggplot will have plots from each sample on a single column. Default TRUE.
sampleRelHeights	If there are multiple samples and combining by "all", the relative heights for each plot.
sampleRelWidths	If there are multiple samples and combining by "all", the relative widths for each plot.

Value

a ggplot object of the scatter plot.

Examples

```
data(scExample, package="singleCellTK")
sce <- runEmptyDrops(inSCE=sce)
plotEmptyDropsScatter(inSCE=sce)
```

plotMarkerDiffExp *Plot a heatmap to visualize the result of* findMarkerDiffExp

Description

This function will first reads the result saved in metadata slot, named by "findMarker" and generated by findMarkerDiffExp. Then it do the filtering on the statistics based on the input parameters and get unique genes to plot. We choose the genes that are identified as up-regulated only. As for the genes identified as up-regulated for multiple clusters, we only keep the belonging towards the one they have the highest Log2FC value. In the heatmap, there will always be a cell annotation for the cluster labeling used when finding the markers, and a feature annotation for which cluster each gene belongs to. And by default we split the heatmap by these two annotations. Additional legends can be added and the splitting can be canceled.

This function will first reads the result saved in metadata slot, named by "findMarker" and generated by findMarkerDiffExp. Then it do the filtering on the statistics based on the input parameters and get unique genes to plot. We choose the genes that are identified as up-regulated only. As for the genes identified as up-regulated for multiple clusters, we only keep the belonging towards the one they have the highest Log2FC value. In the heatmap, there will always be a cell annotation for the cluster labeling used when finding the markers, and a feature annotation for which cluster each gene belongs to. And by default we split the heatmap by these two annotations. Additional legends can be added and the splitting can be canceled.

Usage

```
plotMarkerDiffExp(
  inSCE,
  orderBy = "size",
  log2fcThreshold = 1,
  fdrThreshold = 0.05,
  minClustExprPerc = 0.7,
  maxCtrlExprPerc = 0.4,
  minMeanExpr = 1,
  topN = 10,
  decreasing = TRUE,
  rowDataName = NULL,
  colDataName = NULL,
  featureAnnotations = NULL,
  cellAnnotations = NULL,
  featureAnnotationColor = NULL,
  cellAnnotationColor = NULL,
 colSplitBy = ifelse(is.null(orderBy), NULL, colnames(inSCE@metadata$findMarker)[5]),
  rowSplitBy = "marker",
  rowDend = FALSE,
  colDend = FALSE,
  title = "Top Marker Heatmap",
  ...
```

```
)

plotMarkerDiffExp(
  inSCE,
  orderBy = "size",
  log2fcThreshold = 1,
  fdrThreshold = 0.05,
  minClustExprPerc = 0.7,
  maxCtrlExprPerc = 0.4,
  minMeanExpr = 1,
  topN = 10,
  decreasing = TRUE,
  rowDataName = NULL,
  colDataName = NULL,
  featureAnnotations = NULL,
  cellAnnotations = NULL,
  featureAnnotationColor = NULL,
  cellAnnotationColor = NULL,
 colSplitBy = ifelse(is.null(orderBy), NULL, colnames(inSCE@metadata$findMarker)[5]),
  rowSplitBy = "marker",
  rowDend = FALSE,
  colDend = FALSE,
  title = "Top Marker Heatmap",
  ...
)
```

Arguments

inSCE	SingleCellExperiment inherited object.
orderBy	The ordering method of the clusters on the splitted heatmap. Can be chosen from "size" or "name", specified with vector of ordered unique cluster labels, or set as NULL for unsplitted heatmap. Default "size".
log2fcThreshold	Only use DEGs with the absolute values of log2FC larger than this value. Default 1
fdrThreshold	Only use DEGs with FDR value smaller than this value. Default 0.05
minClustExprPerc	A numeric scalar. The minimum cutoff of the percentage of cells in the cluster of interests that expressed the marker gene. Default 0.7.
maxCtrlExprPerc	A numeric scalar. The maximum cutoff of the percentage of cells out of the cluster (control group) that expressed the marker gene. Default 0.4.
minMeanExpr	A numeric scalar. The minimum cutoff of the mean expression value of the marker in the cluster of interests. Default 1.
topN	An integer. Only to plot this number of top markers for each cluster in maximum, in terms of log2FC value. Use NULL to cancel the top N subscription. Default 10.

decreasing Order the cluster decreasingly. Default TRUE.

rowDataName character. The column name(s) in rowData that need to be added to the annotation. Default NULL.

colDataName character. The column name(s) in colData that need to be added to the annotation. Default NULL.

featureAnnotations
data.frame, with rownames containing all the features going to be plotted. Character columns should be factors. Default NULL.

cellAnnotations
data.frame, with rownames containing all the cells going to be plotted. Character columns should be factors. Default NULL.

featureAnnotationColor
A named list. Customized color settings for feature labeling. Should match the entries in the featureAnnotations or rowDataName. For each entry, there should be a list/vector of colors named with categories. Default NULL.

cellAnnotationColor
A named list. Customized color settings for cell labeling. Should match the entries in the cellAnnotations or colDataName. For each entry, there should be a list/vector of colors named with categories. Default NULL.

colSplitBy character vector. Do semi-heatmap based on the grouping of this(these) annotation(s). Should exist in either colDataName or names(cellAnnotations). Default is the value of cluster in findMarkerDiffExp when orderBy is not NULL, or NULL otherwise.

rowSplitBy character vector. Do semi-heatmap based on the grouping of this(these) annotation(s). Should exist in either rowDataName or names(featureAnnotations). Default "marker", which indicates an auto generated annotation for this plot.

rowDend Whether to display row dendrogram. Default FALSE.

colDend Whether to display column dendrogram. Default FALSE.

title Text of the title, at the top of the heatmap. Default "Top Marker Heatmap".

... Other arguments passed to plotSCEHeatmap.

Value

A Heatmap object

A Heatmap object

Author(s)

Yichen Wang

Yichen Wang

Examples

```
data("sceBatches")
logcounts(sceBatches) <- log(counts(sceBatches) + 1)
sce.w <- subsetSCECols(sceBatches, colData = "batch == 'w'")
```

```
sce.w <- findMarkerDiffExp(sce.w, method = "wilcox", cluster = "cell_type")
plotMarkerDiffExp(sce.w)
data("sceBatches")
logcounts(sceBatches) <- log(counts(sceBatches) + 1)
sce.w <- subsetSCECols(sceBatches, colData = "batch == 'w'")
sce.w <- findMarkerDiffExp(sce.w, method = "wilcox", cluster = "cell_type")
plotMarkerDiffExp(sce.w)
```

`plotMASTThresholdGenes`

MAST Identify adaptive thresholds

Description

Calculate and produce a list of thresholded counts (on natural scale), thresholds, bins, densities estimated on each bin, and the original data from `thresholdSCRNACountMatrix`

Usage

```
plotMASTThresholdGenes(
  inSCE,
  useAssay = "logcounts",
  doPlot = TRUE,
  isLogged = TRUE,
  check_sanity = TRUE
)
```

Arguments

`inSCE`	SingleCellExperiment object
`useAssay`	character, default `"logcounts"`
`doPlot`	Logical scalar. Whether to directly plot in the plotting area. If `FALSE`, will return a graphical object which can be visualized with `grid.draw()`. Default `TRUE`.
`isLogged`	Logical scalar. Whether the assay used for the analysis is logged. If not, will do a `log(assay + 1)` transformation. Default `TRUE`.
`check_sanity`	Logical scalar. Whether to perform MAST's sanity check to see if the counts are logged. Default `TRUE`

Value

Plot the thresholding onto the plotting region if `plot == TRUE` or a graphical object if `plot == FALSE`.

Examples

```
data("mouseBrainSubsetSCE")
plotMASTThresholdGenes(mouseBrainSubsetSCE)
```

plotPCA *Plot PCA run data from its components.*

Description

Plot PCA run data from its components.

Usage

```
plotPCA(
  inSCE,
  colorBy = "No Color",
  shape = "No Shape",
  pcX = "PC1",
  pcY = "PC2",
  reducedDimName = "PCA",
  runPCA = FALSE,
  useAssay = "logcounts"
)
```

Arguments

inSCE	Input SingleCellExperiment object.
colorBy	The variable to color clusters by
shape	Shape of the points
pcX	User choice for the first principal component
pcY	User choice for the second principal component
reducedDimName	a name to store the results of the dimension reduction coordinates obtained from this method. This is stored in the SingleCellExperiment object in the reducedDims slot. Required.
runPCA	Run PCA if the reducedDimName does not exist. the Default is FALSE.
useAssay	Indicate which assay to use. The default is "logcounts".

Value

A PCA plot

Examples

```
data("mouseBrainSubsetSCE")
plotPCA(mouseBrainSubsetSCE, colorBy = "level1class",
        reducedDimName = "PCA_counts")
```

plotRunPerCellQCResults *Plots for runPerCellQC outputs.*

Description

A wrapper function which visualizes outputs from the runPerCellQC function stored in the colData slot of the SingleCellExperiment object via various plots.

Usage

```
plotRunPerCellQCResults(
  inSCE,
  sample = NULL,
  groupBy = NULL,
  combinePlot = "all",
  violin = TRUE,
  boxplot = FALSE,
  dots = TRUE,
  dotSize = 0.5,
  summary = "median",
  summaryTextSize = 3,
  baseSize = 15,
  axisSize = NULL,
  axisLabelSize = NULL,
  transparency = 1,
  defaultTheme = TRUE,
  titleSize = NULL,
  relHeights = 1,
  relWidths = 1,
  labelSamples = TRUE,
  plotNCols = NULL,
  plotNRows = NULL,
  samplePerColumn = TRUE,
  sampleRelHeights = 1,
  sampleRelWidths = 1
)
```

Arguments

inSCE	Input SingleCellExperiment object with saved dimension reduction components or a variable with saved results from runPerCellQC. Required.
sample	Character vector. Indicates which sample each cell belongs to. Default NULL.
groupBy	Groupings for each numeric value. A user may input a vector equal length to the number of the samples in the SingleCellExperiment object, or can be retrieved from the colData slot. Default NULL.

combinePlot	Must be either "all", "sample", or "none". "all" will combine all plots into a single .ggplot object, while "sample" will output a list of plots separated by sample. Default "all".
violin	Boolean. If TRUE, will plot the violin plot. Default TRUE.
boxplot	Boolean. If TRUE, will plot boxplots for each violin plot. Default FALSE.
dots	Boolean. If TRUE, will plot dots for each violin plot. Default TRUE.
dotSize	Size of dots. Default 0.5.
summary	Adds a summary statistic, as well as a crossbar to the violin plot. Options are "mean" or "median". Default "median".
summaryTextSize	The text size of the summary statistic displayed above the violin plot. Default 3.
baseSize	The base font size for all text. Default 15. Can be overwritten by titleSize, axisSize, and axisLabelSize.
axisSize	Size of x/y-axis ticks. Default NULL.
axisLabelSize	Size of x/y-axis labels. Default NULL.
transparency	Transparency of the dots, values will be 0-1. Default 1.
defaultTheme	Removes grid in plot and sets axis title size to 10 when TRUE. Default TRUE.
titleSize	Size of title of plot. Default NULL.
relHeights	Relative heights of plots when combine is set.
relWidths	Relative widths of plots when combine is set.
labelSamples	Will label sample name in title of plot if TRUE. Default TRUE.
plotNCols	Number of columns when plots are combined in a grid.
plotNRows	Number of rows when plots are combined in a grid.
samplePerColumn	If TRUE, when there are multiple samples and combining by "all", the output .ggplot will have plots from each sample on a single column. Default TRUE.
sampleRelHeights	If there are multiple samples and combining by "all", the relative heights for each plot.
sampleRelWidths	If there are multiple samples and combining by "all", the relative widths for each plot.

Value

list of .ggplot objects

Examples

```
data(scExample, package="singleCellTK")
## Not run:
sce <- subsetSCECols(sce, colData = "type != 'EmptyDroplet'")
sce <- runPerCellQC(sce)
plotRunPerCellQCResults(inSCE=sce)

## End(Not run)
```

plotScDblFinderResults *Plots for runScDblFinder outputs.*

Description

A wrapper function which visualizes outputs from the runScDblFinder function stored in the colData slot of the SingleCellExperiment object via various plots.

Usage

```
plotScDblFinderResults(
  inSCE,
  sample = NULL,
  shape = NULL,
  groupBy = NULL,
  combinePlot = "all",
  violin = TRUE,
  boxplot = FALSE,
  dots = TRUE,
  reducedDimName = "UMAP",
  xlab = NULL,
  ylab = NULL,
  dim1 = NULL,
  dim2 = NULL,
  bin = NULL,
  binLabel = NULL,
  defaultTheme = TRUE,
  dotSize = 0.5,
  summary = "median",
  summaryTextSize = 3,
  transparency = 1,
  baseSize = 15,
  titleSize = NULL,
  axisLabelSize = NULL,
  axisSize = NULL,
  legendSize = NULL,
  legendTitleSize = NULL,
  relHeights = 1,
  relWidths = c(1, 1, 1),
  plotNCols = NULL,
  plotNRows = NULL,
  labelSamples = TRUE,
  samplePerColumn = TRUE,
  sampleRelHeights = 1,
  sampleRelWidths = 1
)
```

Arguments

inSCE	Input SingleCellExperiment object with saved dimension reduction components or a variable with saved results from runScDblFinder. Required.
sample	Character vector. Indicates which sample each cell belongs to. Default NULL.
shape	If provided, add shapes based on the value.
groupBy	Groupings for each numeric value. A user may input a vector equal length to the number of the samples in the SingleCellExperiment object, or can be retrieved from the colData slot. Default NULL.
combinePlot	Must be either "all", "sample", or "none". "all" will combine all plots into a single .ggplot object, while "sample" will output a list of plots separated by sample. Default "all".
violin	Boolean. If TRUE, will plot the violin plot. Default TRUE.
boxplot	Boolean. If TRUE, will plot boxplots for each violin plot. Default TRUE.
dots	Boolean. If TRUE, will plot dots for each violin plot. Default TRUE.
reducedDimName	Saved dimension reduction name in the SingleCellExperiment object. Required.
xlab	Character vector. Label for x-axis. Default NULL.
ylab	Character vector. Label for y-axis. Default NULL.
dim1	1st dimension to be used for plotting. Can either be a string which specifies the name of the dimension to be plotted from reducedDims, or a numeric value which specifies the index of the dimension to be plotted. Default is NULL.
dim2	2nd dimension to be used for plotting. Can either be a string which specifies the name of the dimension to be plotted from reducedDims, or a numeric value which specifies the index of the dimension to be plotted. Default is NULL.
bin	Numeric vector. If single value, will divide the numeric values into the 'bin' groups. If more than one value, will bin numeric values using values as a cut point.
binLabel	Character vector. Labels for the bins created by the 'bin' parameter. Default NULL.
defaultTheme	Removes grid in plot and sets axis title size to 10 when TRUE. Default TRUE.
dotSize	Size of dots. Default 0.5.
summary	Adds a summary statistic, as well as a crossbar to the violin plot. Options are "mean" or "median". Default NULL.
summaryTextSize	The text size of the summary statistic displayed above the violin plot. Default 3.
transparency	Transparency of the dots, values will be 0-1. Default 1.
baseSize	The base font size for all text. Default 12. Can be overwritten by titleSize, axisSize, and axisLabelSize, legendSize, legendTitleSize.
titleSize	Size of title of plot. Default NULL.
axisLabelSize	Size of x/y-axis labels. Default NULL.
axisSize	Size of x/y-axis ticks. Default NULL.
legendSize	size of legend. Default NULL.

legendTitleSize
: size of legend title. Default NULL.

relHeights
: Relative heights of plots when combine is set.

relWidths
: Relative widths of plots when combine is set.

plotNCols
: Number of columns when plots are combined in a grid.

plotNRows
: Number of rows when plots are combined in a grid.

labelSamples
: Will label sample name in title of plot if TRUE. Default TRUE.

samplePerColumn
: If TRUE, when there are multiple samples and combining by "all", the output .ggplot will have plots from each sample on a single column. Default TRUE.

sampleRelHeights
: If there are multiple samples and combining by "all", the relative heights for each plot.

sampleRelWidths
: If there are multiple samples and combining by "all", the relative widths for each plot.

Value

list of .ggplot objects

Examples

```
data(scExample, package="singleCellTK")
sce <- subsetSCECols(sce, colData = "type != 'EmptyDroplet'")
sce <- getUMAP(inSCE=sce, useAssay="counts", reducedDimName="UMAP")
sce <- runScDblFinder(sce)
plotScDblFinderResults(inSCE=sce, reducedDimName="UMAP")
```

---

plotScdsHybridResults *Plots for runCxdsBcdsHybrid outputs.*

Description

A wrapper function which visualizes outputs from the runCxdsBcdsHybrid function stored in the colData slot of the SingleCellExperiment object via various plots.

Usage

```
plotScdsHybridResults(
  inSCE,
  sample = NULL,
  shape = NULL,
  groupBy = NULL,
  combinePlot = "all",
  violin = TRUE,
```

```
  boxplot = FALSE,
  dots = TRUE,
  reducedDimName = "UMAP",
  xlab = NULL,
  ylab = NULL,
  dim1 = NULL,
  dim2 = NULL,
  bin = NULL,
  binLabel = NULL,
  defaultTheme = TRUE,
  dotSize = 0.5,
  summary = "median",
  summaryTextSize = 3,
  transparency = 1,
  baseSize = 15,
  titleSize = NULL,
  axisLabelSize = NULL,
  axisSize = NULL,
  legendSize = NULL,
  legendTitleSize = NULL,
  relHeights = 1,
  relWidths = c(1, 1, 1),
  plotNCols = NULL,
  plotNRows = NULL,
  labelSamples = TRUE,
  samplePerColumn = TRUE,
  sampleRelHeights = 1,
  sampleRelWidths = 1
)
```

Arguments

inSCE	Input SingleCellExperiment object with saved dimension reduction components or a variable with saved results from runCxdsBcdsHybrid. Required.
sample	Character vector. Indicates which sample each cell belongs to. Default NULL.
shape	If provided, add shapes based on the value.
groupBy	Groupings for each numeric value. A user may input a vector equal length to the number of the samples in the SingleCellExperiment object, or can be retrieved from the colData slot. Default NULL.
combinePlot	Must be either "all", "sample", or "none". "all" will combine all plots into a single .ggplot object, while "sample" will output a list of plots separated by sample. Default "all".
violin	Boolean. If TRUE, will plot the violin plot. Default TRUE.
boxplot	Boolean. If TRUE, will plot boxplots for each violin plot. Default TRUE.
dots	Boolean. If TRUE, will plot dots for each violin plot. Default TRUE.
reducedDimName	Saved dimension reduction name in the SingleCellExperiment object. Required.

xlab	Character vector. Label for x-axis. Default NULL.
ylab	Character vector. Label for y-axis. Default NULL.
dim1	1st dimension to be used for plotting. Can either be a string which specifies the name of the dimension to be plotted from reducedDims, or a numeric value which specifies the index of the dimension to be plotted. Default is NULL.
dim2	2nd dimension to be used for plotting. Can either be a string which specifies the name of the dimension to be plotted from reducedDims, or a numeric value which specifies the index of the dimension to be plotted. Default is NULL.
bin	Numeric vector. If single value, will divide the numeric values into the 'bin' groups. If more than one value, will bin numeric values using values as a cut point.
binLabel	Character vector. Labels for the bins created by the 'bin' parameter. Default NULL.
defaultTheme	Removes grid in plot and sets axis title size to 10 when TRUE. Default TRUE.
dotSize	Size of dots. Default 0.5.
summary	Adds a summary statistic, as well as a crossbar to the violin plot. Options are "mean" or "median". Default NULL.
summaryTextSize	The text size of the summary statistic displayed above the violin plot. Default 3.
transparency	Transparency of the dots, values will be 0-1. Default 1.
baseSize	The base font size for all text. Default 12. Can be overwritten by titleSize, axisSize, and axisLabelSize, legendSize, legendTitleSize.
titleSize	Size of title of plot. Default NULL.
axisLabelSize	Size of x/y-axis labels. Default NULL.
axisSize	Size of x/y-axis ticks. Default NULL.
legendSize	size of legend. Default NULL.
legendTitleSize	size of legend title. Default NULL.
relHeights	Relative heights of plots when combine is set.
relWidths	Relative widths of plots when combine is set.
plotNCols	Number of columns when plots are combined in a grid.
plotNRows	Number of rows when plots are combined in a grid.
labelSamples	Will label sample name in title of plot if TRUE. Default TRUE.
samplePerColumn	If TRUE, when there are multiple samples and combining by "all", the output .ggplot will have plots from each sample on a single column. Default TRUE.
sampleRelHeights	If there are multiple samples and combining by "all", the relative heights for each plot.
sampleRelWidths	If there are multiple samples and combining by "all", the relative widths for each plot.

Value

list of .ggplot objects

Examples

```
data(scExample, package="singleCellTK")
sce <- subsetSCECols(sce, colData = "type != 'EmptyDroplet'")
sce <- getUMAP(inSCE=sce, useAssay="counts", reducedDimName="UMAP")
sce <- runCxdsBcdsHybrid(sce)
plotScdsHybridResults(inSCE=sce, reducedDimName="UMAP")
```

---

plotSCEBarAssayData *Bar plot of assay data.*

Description

Visualizes values stored in the assay slot of a SingleCellExperiment object via a bar plot.

Usage

```
plotSCEBarAssayData(
  inSCE,
  feature,
  sample = NULL,
  useAssay = "counts",
  featureLocation = NULL,
  featureDisplay = NULL,
  groupBy = NULL,
  xlab = NULL,
  ylab = NULL,
  axisSize = 10,
  axisLabelSize = 10,
  dotSize = 0.5,
  transparency = 1,
  defaultTheme = TRUE,
  gridLine = FALSE,
  summary = NULL,
  title = NULL,
  titleSize = NULL,
  combinePlot = TRUE
)
```

Arguments

inSCE	Input SingleCellExperiment object with saved dimension reduction components or a variable with saved results. Required.
feature	Name of feature stored in assay of SingleCellExperiment object.

sample	Character vector. Indicates which sample each cell belongs to.
useAssay	Indicate which assay to use. Default "counts".
featureLocation	Indicates which column name of rowData to query gene.
featureDisplay	Indicates which column name of rowData to use to display feature for visualization.
groupBy	Groupings for each numeric value. A user may input a vector equal length to the number of the samples in the SingleCellExperiment object, or can be retrieved from the colData slot. Default NULL.
xlab	Character vector. Label for x-axis. Default NULL.
ylab	Character vector. Label for y-axis. Default NULL.
axisSize	Size of x/y-axis ticks. Default 10.
axisLabelSize	Size of x/y-axis labels. Default 10.
dotSize	Size of dots. Default 0.5.
transparency	Transparency of the dots, values will be 0-1. Default 1.
defaultTheme	Removes grid in plot and sets axis title size to 10 when TRUE. Default TRUE.
gridLine	Adds a horizontal grid line if TRUE. Will still be drawn even if defaultTheme is TRUE. Default FALSE.
summary	Adds a summary statistic, as well as a crossbar to the violin plot. Options are "mean" or "median". Default NULL.
title	Title of plot. Default NULL.
titleSize	Size of title of plot. Default 15.
combinePlot	Boolean. If multiple plots are generated (multiple samples, etc.), will combined plots using ‘cowplot::plot_grid‘. Default TRUE.

Value

a ggplot of the barplot of assay data.

Examples

```
plotSCEBarAssayData(
  inSCE = mouseBrainSubsetSCE,
  feature = "Apoe", groupBy = "sex"
)
```

plotSCEBarColData *Bar plot of colData.*

Description

Visualizes values stored in the colData slot of a SingleCellExperiment object via a bar plot.

Usage

```
plotSCEBarColData(
  inSCE,
  coldata,
  sample = NULL,
  groupBy = NULL,
  dots = TRUE,
  xlab = NULL,
  ylab = NULL,
  axisSize = 10,
  axisLabelSize = 10,
  dotSize = 0.5,
  transparency = 1,
  defaultTheme = TRUE,
  gridLine = FALSE,
  summary = NULL,
  title = NULL,
  titleSize = NULL,
  combinePlot = TRUE
)
```

Arguments

inSCE	Input SingleCellExperiment object with saved dimension reduction components or a variable with saved results. Required.
coldata	colData value that will be plotted.
sample	Character vector. Indicates which sample each cell belongs to.
groupBy	Groupings for each numeric value. A user may input a vector equal length to the number of the samples in the SingleCellExperiment object, or can be retrieved from the colData slot. Default NULL.
dots	Boolean. If TRUE, will plot dots for each violin plot. Default TRUE.
xlab	Character vector. Label for x-axis. Default NULL.
ylab	Character vector. Label for y-axis. Default NULL.
axisSize	Size of x/y-axis ticks. Default 10.
axisLabelSize	Size of x/y-axis labels. Default 10.
dotSize	Size of dots. Default 0.5.

transparency	Transparency of the dots, values will be 0-1. Default 1.
defaultTheme	Removes grid in plot and sets axis title size to 10 when TRUE. Default TRUE.
gridLine	Adds a horizontal grid line if TRUE. Will still be drawn even if defaultTheme is TRUE. Default FALSE.
summary	Adds a summary statistic, as well as a crossbar to the violin plot. Options are "mean" or "median". Default NULL.
title	Title of plot. Default NULL.
titleSize	Size of title of plot. Default 15.
combinePlot	Boolean. If multiple plots are generated (multiple samples, etc.), will combined plots using ‘cowplot::plot_grid‘. Default TRUE.

Value

a ggplot of the barplot of coldata.

Examples

```
plotSCEBarColData(
  inSCE = mouseBrainSubsetSCE,
  coldata = "age", groupBy = "sex"
)
```

---

plotSCEBatchFeatureMean

Plot mean feature value in each batch of a SingleCellExperiment object

---

Description

Plot mean feature value in each batch of a SingleCellExperiment object

Usage

```
plotSCEBatchFeatureMean(
  inSCE,
  useAssay = NULL,
  useReddim = NULL,
  useAltExp = NULL,
  batch = "batch",
  xlab = "batch",
  ylab = "Feature Mean",
  ...
)
```

Arguments

inSCE	SingleCellExperiment inherited object.
useAssay	A single character. The name of the assay that stores the value to plot. For useReddim and useAltExp also. Default NULL.
useReddim	A single character. The name of the dimension reduced matrix that stores the value to plot. Default NULL.
useAltExp	A single character. The name of the alternative experiment that stores an assay of the value to plot. Default NULL.
batch	A single character. The name of batch annotation column in colData(inSCE). Default "batch".
xlab	label for x-axis. Default "batch".
ylab	label for y-axis. Default "Feature Mean".
...	Additional arguments passed to .ggViolin.

Value

ggplot

Examples

```
data('sceBatches', package = 'singleCellTK')
plotSCEBatchFeatureMean(sceBatches, useAssay = "counts")
```

---

plotSCEDensity *Density plot of any data stored in the SingleCellExperiment object.*

Description

Visualizes values stored in any slot of a SingleCellExperiment object via a densityn plot.

Usage

```
plotSCEDensity(
  inSCE,
  slotName,
  itemName,
  sample = NULL,
  feature = NULL,
  dimension = NULL,
  groupBy = NULL,
  xlab = NULL,
  ylab = NULL,
  axisSize = 10,
  axisLabelSize = 10,
  defaultTheme = TRUE,
```

```
  title = NULL,
  titleSize = 18,
  cutoff = NULL,
  combinePlot = "none",
  plotLabels = NULL
)
```

Arguments

inSCE	Input SingleCellExperiment object with saved dimension reduction components or a variable with saved results. Required.
slotName	Desired slot of SingleCellExperiment used for plotting. Possible options: "assays", "colData", "metadata", "reducedDims". Required.
itemName	Desired vector within the slot used for plotting. Required.
sample	Character vector. Indicates which sample each cell belongs to.
feature	Desired name of feature stored in assay of SingleCellExperiment object. Only used when "assays" slotName is selected. Default NULL.
dimension	Desired dimension stored in the specified reducedDims. Either an integer which indicates the column or a character vector specifies column name. By default, the 1st dimension/column will be used. Only used when "reducedDims" slotName is selected. Default NULL.
groupBy	Groupings for each numeric value. A user may input a vector equal length to the number of the samples in the SingleCellExperiment object, or can be retrieved from the colData slot. Default NULL.
xlab	Character vector. Label for x-axis. Default NULL.
ylab	Character vector. Label for y-axis. Default NULL.
axisSize	Size of x/y-axis ticks. Default 10.
axisLabelSize	Size of x/y-axis labels. Default 10.
defaultTheme	Removes grid in plot and sets axis title size to 10 when TRUE. Default TRUE.
title	Title of plot. Default NULL.
titleSize	Size of title of plot. Default 15.
cutoff	Numeric value. The plot will be annotated with a vertical line if set. Default NULL.
combinePlot	Must be either "all", "sample", or "none". "all" will combine all plots into a single .ggplot object, while "sample" will output a list of plots separated by sample. Default "none".
plotLabels	labels to each plot. If set to "default", will use the name of the samples as the labels. If set to "none", no label will be plotted.

Value

a ggplot object of the density plot.

Examples

```
plotSCEDensity(
  inSCE = mouseBrainSubsetSCE, slotName = "assays",
  itemName = "counts", feature = "Apoe", groupBy = "sex"
)
```

`plotSCEDensityAssayData` *Density plot of assay data.*

Description

Visualizes values stored in the assay slot of a SingleCellExperiment object via a density plot.

Usage

```
plotSCEDensityAssayData(
  inSCE,
  feature,
  sample = NULL,
  useAssay = "counts",
  featureLocation = NULL,
  featureDisplay = NULL,
  groupBy = NULL,
  xlab = NULL,
  ylab = NULL,
  axisSize = 10,
  axisLabelSize = 10,
  defaultTheme = TRUE,
  cutoff = NULL,
  title = NULL,
  titleSize = 18,
  combinePlot = "none",
  plotLabels = NULL
)
```

Arguments

`inSCE`	Input SingleCellExperiment object with saved dimension reduction components or a variable with saved results. Required.
`feature`	Name of feature stored in assay of SingleCellExperiment object.
`sample`	Character vector. Indicates which sample each cell belongs to.
`useAssay`	Indicate which assay to use. Default "counts".
`featureLocation`	Indicates which column name of rowData to query gene.

featureDisplay	Indicates which column name of rowData to use to display feature for visualization.
groupBy	Groupings for each numeric value. A user may input a vector equal length to the number of the samples in the SingleCellExperiment object, or can be retrieved from the colData slot. Default NULL.
xlab	Character vector. Label for x-axis. Default NULL.
ylab	Character vector. Label for y-axis. Default NULL.
axisSize	Size of x/y-axis ticks. Default 10.
axisLabelSize	Size of x/y-axis labels. Default 10.
defaultTheme	Removes grid in plot and sets axis title size to 10 when TRUE. Default TRUE.
cutoff	Numeric value. The plot will be annotated with a vertical line if set. Default NULL.
title	Title of plot. Default NULL.
titleSize	Size of title of plot. Default 15.
combinePlot	Must be either "all", "sample", or "none". "all" will combine all plots into a single .ggplot object, while "sample" will output a list of plots separated by sample. Default "none".
plotLabels	labels to each plot. If set to "default", will use the name of the samples as the labels. If set to "none", no label will be plotted.

Value

a ggplot of the density plot of assay data.

Examples

```
plotSCEDensityAssayData(
  inSCE = mouseBrainSubsetSCE,
  feature = "Apoe"
)
```

---

plotSCEDensityColData *Density plot of colData.*

Description

Visualizes values stored in the colData slot of a SingleCellExperiment object via a density plot.

Usage

```
plotSCEDensityColData(
  inSCE,
  coldata,
  sample = NULL,
  groupBy = NULL,
  xlab = NULL,
  ylab = NULL,
  baseSize = 12,
  axisSize = NULL,
  axisLabelSize = NULL,
  defaultTheme = TRUE,
  title = NULL,
  titleSize = 18,
  cutoff = NULL,
  combinePlot = "none",
  plotLabels = NULL
)
```

Arguments

inSCE	Input SingleCellExperiment object with saved dimension reduction components or a variable with saved results. Required.
coldata	colData value that will be plotted.
sample	Character vector. Indicates which sample each cell belongs to.
groupBy	Groupings for each numeric value. A user may input a vector equal length to the number of the samples in the SingleCellExperiment object, or can be retrieved from the colData slot. Default NULL.
xlab	Character vector. Label for x-axis. Default NULL.
ylab	Character vector. Label for y-axis. Default NULL.
baseSize	The base font size for all text. Default 12. Can be overwritten by titleSize, axisSize, and axisLabelSize, legendSize, legendTitleSize.
axisSize	Size of x/y-axis ticks. Default NULL.
axisLabelSize	Size of x/y-axis labels. Default NULL.
defaultTheme	Removes grid in plot and sets axis title size to 10 when TRUE. Default TRUE.
title	Title of plot. Default NULL.
titleSize	Size of title of plot. Default 15.
cutoff	Numeric value. The plot will be annotated with a vertical line if set. Default NULL.
combinePlot	Must be either "all", "sample", or "none". "all" will combine all plots into a single .ggplot object, while "sample" will output a list of plots separated by sample. Default "none".
plotLabels	labels to each plot. If set to "default", will use the name of the samples as the labels. If set to "none", no label will be plotted.

Value

a ggplot of the density plot of colData.

Examples

```
plotSCEDensityColData(
  inSCE = mouseBrainSubsetSCE,
  coldata = "age", groupBy = "sex"
)
```

---

plotSCEDimReduceColData

Dimension reduction plot tool for colData

Description

Plot results of reduced dimensions data and colors by annotation data stored in the colData slot.

Usage

```
plotSCEDimReduceColData(
  inSCE,
  colorBy,
  reducedDimName,
  sample = NULL,
  groupBy = NULL,
  conditionClass = NULL,
  shape = NULL,
  xlab = NULL,
  ylab = NULL,
  baseSize = 12,
  axisSize = NULL,
  axisLabelSize = NULL,
  dim1 = NULL,
  dim2 = NULL,
  bin = NULL,
  binLabel = NULL,
  dotSize = 0.5,
  transparency = 1,
  colorScale = NULL,
  colorLow = "white",
  colorMid = "gray",
  colorHigh = "blue",
  defaultTheme = TRUE,
  title = NULL,
  titleSize = 15,
  labelClusters = TRUE,
```

```
  clusterLabelSize = 3.5,
  legendTitle = NULL,
  legendTitleSize = NULL,
  legendSize = NULL,
  combinePlot = "none",
  plotLabels = NULL
)
```

Arguments

inSCE	Input SingleCellExperiment object with saved dimension reduction components or a variable with saved results. Required.
colorBy	Color by a condition(any column of the annotation data). Required.
reducedDimName	Saved dimension reduction matrix name in the SingleCellExperiment object. Required.
sample	Character vector. Indicates which sample each cell belongs to.
groupBy	Group by a condition(any column of the annotation data). Default NULL.
conditionClass	Class of the annotation data used in colorBy. Options are NULL, "factor" or "numeric". If NULL, class will default to the original class. Default NULL.
shape	Add shapes to each condition.
xlab	Character vector. Label for x-axis. Default NULL.
ylab	Character vector. Label for y-axis. Default NULL.
baseSize	The base font size for all text. Default 12. Can be overwritten by titleSize, axisSize, and axisLabelSize, legendSize, legendTitleSize.
axisSize	Size of x/y-axis ticks. Default NULL.
axisLabelSize	Size of x/y-axis labels. Default NULL.
dim1	1st dimension to be used for plotting. Can either be a string which specifies the name of the dimension to be plotted from reducedDims, or a numeric value which specifies the index of the dimension to be plotted. Default is NULL.
dim2	2nd dimension to be used for plotting. Can either be a string which specifies the name of the dimension to be plotted from reducedDims, or a numeric value which specifies the index of the dimension to be plotted. Default is NULL.
bin	Numeric vector. If single value, will divide the numeric values into the 'bin' groups. If more than one value, will bin numeric values using values as a cut point.
binLabel	Character vector. Labels for the bins created by the 'bin' parameter. Default NULL.
dotSize	Size of dots. Default 0.5.
transparency	Transparency of the dots, values will be 0-1. Default 1.
colorScale	Vector. Needs to be same length as the number of unique levels of colorBy. Will be used only if conditionClass = "factor" or "character". Default NULL.
colorLow	Character. A color available from 'colors()'. The color will be used to signify the lowest values on the scale. Default 'white'.

colorMid	Character. A color available from 'colors()'. The color will be used to signify the midpoint on the scale. Default 'gray'.
colorHigh	Character. A color available from 'colors()'. The color will be used to signify the highest values on the scale. Default 'blue'.
defaultTheme	adds grid to plot when TRUE. Default TRUE.
title	Title of plot. Default NULL.
titleSize	Size of title of plot. Default 15.
labelClusters	Logical. Whether the cluster labels are plotted.
clusterLabelSize	Numeric. Determines the size of cluster label when 'labelClusters' is set to TRUE. Default 3.5.
legendTitle	title of legend. Default NULL.
legendTitleSize	size of legend title. Default 12.
legendSize	size of legend. Default NULL. Default FALSE.
combinePlot	Must be either "all", "sample", or "none". "all" will combine all plots into a single .ggplot object, while "sample" will output a list of plots separated by sample. Default "none".
plotLabels	labels to each plot. If set to "default", will use the name of the samples as the labels. If set to "none", no label will be plotted.

Value

a ggplot of the reduced dimension plot of coldata.

Examples

```
plotSCEDimReduceColData(
  inSCE = mouseBrainSubsetSCE, colorBy = "tissue",
  shape = NULL, conditionClass = "factor",
  reducedDimName = "TSNE_counts",
  xlab = "tSNE1", ylab = "tSNE2", labelClusters = TRUE
)

plotSCEDimReduceColData(
  inSCE = mouseBrainSubsetSCE, colorBy = "age",
  shape = NULL, conditionClass = "numeric",
  reducedDimName = "TSNE_counts", bin = c(-Inf, 20, 25, +Inf),
  xlab = "tSNE1", ylab = "tSNE2", labelClusters = FALSE
)
```

plotSCEDimReduceFeatures

Dimension reduction plot tool for assay data

Description

Plot results of reduced dimensions data and colors by feature data stored in the assays slot.

Usage

```
plotSCEDimReduceFeatures(
  inSCE,
  feature,
  reducedDimName,
  sample = NULL,
  featureLocation = NULL,
  featureDisplay = NULL,
  shape = NULL,
  useAssay = "logcounts",
  xlab = NULL,
  ylab = NULL,
  axisSize = 10,
  axisLabelSize = 10,
  dim1 = NULL,
  dim2 = NULL,
  bin = NULL,
  binLabel = NULL,
  dotSize = 0.5,
  transparency = 1,
  colorLow = "white",
  colorMid = "gray",
  colorHigh = "blue",
  defaultTheme = TRUE,
  title = NULL,
  titleSize = 15,
  legendTitle = NULL,
  legendSize = 10,
  legendTitleSize = 12,
  groupBy = NULL,
  combinePlot = "none",
  plotLabels = NULL
)
```

Arguments

inSCE	Input SingleCellExperiment object with saved dimension reduction components or a variable with saved results. Required.

feature	Name of feature stored in assay of SingleCellExperiment object.
reducedDimName	saved dimension reduction name in the SingleCellExperiment object. Required.
sample	Character vector. Indicates which sample each cell belongs to.
featureLocation	Indicates which column name of rowData to query gene.
featureDisplay	Indicates which column name of rowData to use to display feature for visualization.
shape	add shapes to each condition. Default NULL.
useAssay	Indicate which assay to use. The default is "logcounts"
xlab	Character vector. Label for x-axis. Default NULL.
ylab	Character vector. Label for y-axis. Default NULL.
axisSize	Size of x/y-axis ticks. Default 10.
axisLabelSize	Size of x/y-axis labels. Default 10.
dim1	1st dimension to be used for plotting. Can either be a string which specifies the name of the dimension to be plotted from reducedDims, or a numeric value which specifies the index of the dimension to be plotted. Default is NULL.
dim2	2nd dimension to be used for plotting. Can either be a string which specifies the name of the dimension to be plotted from reducedDims, or a numeric value which specifies the index of the dimension to be plotted. Default is NULL.
bin	Numeric vector. If single value, will divide the numeric values into the 'bin' groups. If more than one value, will bin numeric values using values as a cut point.
binLabel	Character vector. Labels for the bins created by the 'bin' parameter. Default NULL.
dotSize	Size of dots. Default 0.5.
transparency	Transparency of the dots, values will be 0-1. Default 1.
colorLow	Character. A color available from 'colors()'. The color will be used to signify the lowest values on the scale. Default 'white'.
colorMid	Character. A color available from 'colors()'. The color will be used to signify the midpoint on the scale. Default 'gray'.
colorHigh	Character. A color available from 'colors()'. The color will be used to signify the highest values on the scale. Default 'blue'.
defaultTheme	adds grid to plot when TRUE. Default TRUE.
title	Title of plot. Default NULL.
titleSize	Size of title of plot. Default 15.
legendTitle	title of legend. Default NULL.
legendSize	size of legend. Default 10.
legendTitleSize	size of legend title. Default 12.
groupBy	Facet wrap the scatterplot based on value. Default NULL.

combinePlot	Must be either "all", "sample", or "none". "all" will combine all plots into a single .ggplot object, while "sample" will output a list of plots separated by sample. Default "none".
plotLabels	labels to each plot. If set to "default", will use the name of the samples as the labels. If set to "none", no label will be plotted.

Value

a ggplot of the reduced dimension plot of feature data.

Examples

```
plotSCEDimReduceFeatures(
  inSCE = mouseBrainSubsetSCE, feature = "Apoe",
  shape = NULL, reducedDimName = "TSNE_counts",
  useAssay = "counts", xlab = "tSNE1", ylab = "tSNE2"
)
```

---

plotSCEHeatmap *Plot heatmap of using data stored in SingleCellExperiment Object*

Description

Plot heatmap of using data stored in SingleCellExperiment Object

Usage

```
plotSCEHeatmap(
  inSCE,
  useAssay = "logcounts",
  doLog = FALSE,
  featureIndex = NULL,
  cellIndex = NULL,
  scale = TRUE,
  trim = c(-2, 2),
  featureIndexBy = "rownames",
  cellIndexBy = "rownames",
  rowDataName = NULL,
  colDataName = NULL,
  featureAnnotations = NULL,
  cellAnnotations = NULL,
  featureAnnotationColor = NULL,
  cellAnnotationColor = NULL,
  rowSplitBy = NULL,
  colSplitBy = NULL,
  rowLabel = FALSE,
  colLabel = FALSE,
```

```
  rowLabelSize = 8,
  colLabelSize = 8,
  rowDend = TRUE,
  colDend = TRUE,
  title = "SCE Heatmap",
  rowTitle = "Genes",
  colTitle = "Cells",
  rowGap = grid::unit(0, "mm"),
  colGap = grid::unit(0, "mm"),
  border = FALSE,
  colorScheme = NULL,
  ...
)
```

Arguments

inSCE	SingleCellExperiment inherited object.
useAssay	character. A string indicating the assay name that provides the expression level to plot.
doLog	Logical scalar. Whether to do log(assay + 1) transformation on the assay indicated by useAssay. Default FALSE.
featureIndex	A vector that can subset the input SCE object by rows (features). Alternatively, it can be a vector identifying features in another feature list indicated by featureIndexBy. Default NULL.
cellIndex	A vector that can subset the input SCE object by columns (cells). Alternatively, it can be a vector identifying cells in another cell list indicated by featureIndexBy. Default NULL.
scale	Whether to perform z-score scaling on each row. Default TRUE.
trim	A 2-element numeric vector. Values outside of this range will be trimmed to their nearst bound. Default c(-2,2)
featureIndexBy	A single character specifying a column name of rowData(inSCE), or a vector of the same length as nrow(inSCE), where we search for the non-rowname feature indices. Default "rownames".
cellIndexBy	A single character specifying a column name of colData(inSCE), or a vector of the same length as ncol(inSCE), where we search for the non-rowname cell indices. Default "rownames".
rowDataName	character. The column name(s) in rowData that need to be added to the annotation. Default NULL.
colDataName	character. The column name(s) in colData that need to be added to the annotation. Default NULL.
featureAnnotations	data.frame, with rownames containing all the features going to be plotted. Character columns should be factors. Default NULL.
cellAnnotations	data.frame, with rownames containing all the cells going to be plotted. Character columns should be factors. Default NULL.

featureAnnotationColor
: A named list. Customized color settings for feature labeling. Should match the entries in the featureAnnotations or rowDataName. For each entry, there should be a list/vector of colors named with categories. Default NULL.

cellAnnotationColor
: A named list. Customized color settings for cell labeling. Should match the entries in the cellAnnotations or colDataName. For each entry, there should be a list/vector of colors named with categories. Default NULL.

rowSplitBy
: character. Do semi-heatmap based on the grouping of this(these) annotation(s). Should exist in either rowDataName or names(featureAnnotations). Default NULL.

colSplitBy
: character. Do semi-heatmap based on the grouping of this(these) annotation(s). Should exist in either colDataName or names(cellAnnotations). Default NULL.

rowLabel
: Use a logical for whether to display all the feature names, a single character to display a column of rowData(inSCE) annotation, a vector of the same length as full/subset nrow(inSCE) to display customized info. Default FALSE.

colLabel
: Use a logical for whether to display all the cell names, a single character to display a column of colData(inSCE) annotation, a vector of the same length as full/subset ncol(inSCE) to display customized info. Default FALSE.

rowLabelSize
: A number for the font size of feature names. Default 8

colLabelSize
: A number for the font size of cell names. Default 8

rowDend
: Whether to display row dendrogram. Default TRUE.

colDend
: Whether to display column dendrogram. Default TRUE.

title
: The main title of the whole plot. Default "SCE Heatmap"

rowTitle
: The subtitle for the rows. Default "Genes".

colTitle
: The subtitle for the columns. Default "Cells".

rowGap
: A numeric value or a unit object. For the gap size between rows of the splitted heatmap. Default grid::unit(0,'mm').

colGap
: A numeric value or a unit object. For the gap size between columns of the splitted heatmap. Default grid::unit(0,'mm').

border
: A logical scalar. Whether to show the border of the heatmap or splitted heatmaps. Default TRUE.

colorScheme
: function. A function that generates color code by giving a value. Can be generated by colorRamp2. Default NULL.

...
: Other arguments passed to Heatmap.

Value

A Heatmap object

Author(s)

Yichen Wang

Examples

```
data(scExample, package = "singleCellTK")
plotSCEHeatmap(sce[1:3,1:3], useAssay = "counts")
```

`plotSCEScatter` *Dimension reduction plot tool for all types of data*

Description

Plot results of reduced dimensions data of counts stored in any slot in the SingleCellExperiment object.

Usage

```
plotSCEScatter(
  inSCE,
  annotation,
  reducedDimName = NULL,
  slot = NULL,
  sample = NULL,
  feature = NULL,
  groupBy = NULL,
  shape = NULL,
  conditionClass = NULL,
  xlab = NULL,
  ylab = NULL,
  axisSize = 10,
  axisLabelSize = 10,
  dim1 = NULL,
  dim2 = NULL,
  bin = NULL,
  binLabel = NULL,
  dotSize = 0.5,
  transparency = 1,
  colorLow = "white",
  colorMid = "gray",
  colorHigh = "blue",
  defaultTheme = TRUE,
  title = NULL,
  titleSize = 15,
  labelClusters = TRUE,
  legendTitle = NULL,
  legendTitleSize = 12,
  legendSize = 10,
  combinePlot = "none",
  plotLabels = NULL
)
```

Arguments

inSCE	Input SingleCellExperiment object with saved dimension reduction components or a variable with saved results. Required.
annotation	Desired vector within the slot used for plotting. Default NULL.
reducedDimName	saved dimension reduction name in the SingleCellExperiment object.
slot	Desired slot of SingleCellExperiment used for plotting. Possible options: "assays", "colData", "metadata", "reducedDims". Default NULL.
sample	Character vector. Indicates which sample each cell belongs to.
feature	name of feature stored in assay of SingleCellExperiment object. Will be used only if "assays" slot is chosen. Default NULL.
groupBy	Group by a condition(any column of the annotation data). Default NULL.
shape	add shapes to each condition.
conditionClass	class of the annotation data used in colorBy. Options are NULL, "factor" or "numeric". If NULL, class will default to the original class. Default NULL.
xlab	Character vector. Label for x-axis. Default NULL.
ylab	Character vector. Label for y-axis. Default NULL.
axisSize	Size of x/y-axis ticks. Default 10.
axisLabelSize	Size of x/y-axis labels. Default 10.
dim1	1st dimension to be used for plotting. Can either be a string which specifies the name of the dimension to be plotted from reducedDims, or a numeric value which specifies the index of the dimension to be plotted. Default is NULL.
dim2	2nd dimension to be used for plotting. Can either be a string which specifies the name of the dimension to be plotted from reducedDims, or a numeric value which specifies the index of the dimension to be plotted. Default is NULL.
bin	Numeric vector. If single value, will divide the numeric values into the ‘bin‘ groups. If more than one value, will bin numeric values using values as a cut point.
binLabel	Character vector. Labels for the bins created by the ‘bin‘ parameter. Default NULL.
dotSize	Size of dots. Default 0.5.
transparency	Transparency of the dots, values will be 0-1. Default 1.
colorLow	Character. A color available from ‘colors()‘. The color will be used to signify the lowest values on the scale. Default ’white’.
colorMid	Character. A color available from ‘colors()‘. The color will be used to signify the midpoint on the scale. Default ’gray’.
colorHigh	Character. A color available from ‘colors()‘. The color will be used to signify the highest values on the scale. Default ’blue’.
defaultTheme	adds grid to plot when TRUE. Default TRUE.
title	Title of plot. Default NULL.
titleSize	Size of title of plot. Default 15.

labelClusters	Logical. Whether the cluster labels are plotted.
legendTitle	title of legend. Default NULL.
legendTitleSize	size of legend title. Default 12.
legendSize	size of legend. Default 10.
combinePlot	Must be either "all", "sample", or "none". "all" will combine all plots into a single .ggplot object, while "sample" will output a list of plots separated by sample. Default "none".
plotLabels	labels to each plot. If set to "default", will use the name of the samples as the labels. If set to "none", no label will be plotted.

Value

a ggplot of the reduced dimensions.

Examples

```
plotSCEScatter(
  inSCE = mouseBrainSubsetSCE, legendTitle = NULL,
  slot = "assays", annotation = "counts", feature = "Apoe",
  reducedDimName = "TSNE_counts", labelClusters = FALSE
)
```

---

plotSCEViolin *Violin plot of any data stored in the SingleCellExperiment object.*

Description

Visualizes values stored in any slot of a SingleCellExperiment object via a violin plot.

Usage

```
plotSCEViolin(
  inSCE,
  slotName,
  itemName,
  feature = NULL,
  sample = NULL,
  dimension = NULL,
  groupBy = NULL,
  violin = TRUE,
  boxplot = TRUE,
  dots = TRUE,
  xlab = NULL,
  ylab = NULL,
  axisSize = 10,
```

```
  axisLabelSize = 10,
  dotSize = 0.5,
  transparency = 1,
  defaultTheme = TRUE,
  gridLine = FALSE,
  summary = NULL,
  title = NULL,
  titleSize = NULL,
  combinePlot = "none",
  plotLabels = NULL
)
```

Arguments

inSCE	Input SingleCellExperiment object with saved dimension reduction components or a variable with saved results. Required.
slotName	Desired slot of SingleCellExperiment used for plotting. Possible options: "assays", "colData", "metadata", "reducedDims". Required.
itemName	Desired vector within the slot used for plotting. Required.
feature	Desired name of feature stored in assay of SingleCellExperiment object. Only used when "assays" slotName is selected. Default NULL.
sample	Character vector. Indicates which sample each cell belongs to.
dimension	Desired dimension stored in the specified reducedDims. Either an integer which indicates the column or a character vector specifies column name. By default, the 1st dimension/column will be used. Only used when "reducedDims" slotName is selected. Default NULL.
groupBy	Groupings for each numeric value. A user may input a vector equal length to the number of the samples in the SingleCellExperiment object, or can be retrieved from the colData slot. Default NULL.
violin	Boolean. If TRUE, will plot the violin plot. Default TRUE.
boxplot	Boolean. If TRUE, will plot boxplots for each violin plot. Default TRUE.
dots	Boolean. If TRUE, will plot dots for each violin plot. Default TRUE.
xlab	Character vector. Label for x-axis. Default NULL.
ylab	Character vector. Label for y-axis. Default NULL.
axisSize	Size of x/y-axis ticks. Default 10.
axisLabelSize	Size of x/y-axis labels. Default 10.
dotSize	Size of dots. Default 0.5.
transparency	Transparency of the dots, values will be 0-1. Default 1.
defaultTheme	Removes grid in plot and sets axis title size to 10 when TRUE. Default TRUE.
gridLine	Adds a horizontal grid line if TRUE. Will still be drawn even if defaultTheme is TRUE. Default FALSE.
summary	Adds a summary statistic, as well as a crossbar to the violin plot. Options are "mean" or "median". Default NULL.

title	Title of plot. Default NULL.
titleSize	Size of title of plot. Default 15.
combinePlot	Must be either "all", "sample", or "none". "all" will combine all plots into a single .ggplot object, while "sample" will output a list of plots separated by sample. Default "none".
plotLabels	labels to each plot. If set to "default", will use the name of the samples as the labels. If set to "none", no label will be plotted.

Value

a ggplot of the violin plot.

Examples

```
plotSCEViolin(
  inSCE = mouseBrainSubsetSCE, slotName = "assays",
  itemName = "counts", feature = "Apoe", groupBy = "sex"
)
```

---

plotSCEViolinAssayData

Violin plot of assay data.

Description

Visualizes values stored in the assay slot of a SingleCellExperiment object via a violin plot.

Usage

```
plotSCEViolinAssayData(
  inSCE,
  feature,
  sample = NULL,
  useAssay = "counts",
  featureLocation = NULL,
  featureDisplay = NULL,
  groupBy = NULL,
  violin = TRUE,
  boxplot = TRUE,
  dots = TRUE,
  xlab = NULL,
  ylab = NULL,
  axisSize = 10,
  axisLabelSize = 10,
  dotSize = 0.5,
  transparency = 1,
  defaultTheme = TRUE,
```

```
  gridLine = FALSE,
  summary = NULL,
  title = NULL,
  titleSize = NULL,
  combinePlot = "none",
  plotLabels = NULL
)
```

Arguments

inSCE	Input SingleCellExperiment object with saved dimension reduction components or a variable with saved results. Required.
feature	Name of feature stored in assay of SingleCellExperiment object.
sample	Character vector. Indicates which sample each cell belongs to.
useAssay	Indicate which assay to use. Default "counts".
featureLocation	Indicates which column name of rowData to query gene.
featureDisplay	Indicates which column name of rowData to use to display feature for visualization.
groupBy	Groupings for each numeric value. A user may input a vector equal length to the number of the samples in the SingleCellExperiment object, or can be retrieved from the colData slot. Default NULL.
violin	Boolean. If TRUE, will plot the violin plot. Default TRUE.
boxplot	Boolean. If TRUE, will plot boxplots for each violin plot. Default TRUE.
dots	Boolean. If TRUE, will plot dots for each violin plot. Default TRUE.
xlab	Character vector. Label for x-axis. Default NULL.
ylab	Character vector. Label for y-axis. Default NULL.
axisSize	Size of x/y-axis ticks. Default 10.
axisLabelSize	Size of x/y-axis labels. Default 10.
dotSize	Size of dots. Default 0.5.
transparency	Transparency of the dots, values will be 0-1. Default 1.
defaultTheme	Removes grid in plot and sets axis title size to 10 when TRUE. Default TRUE.
gridLine	Adds a horizontal grid line if TRUE. Will still be drawn even if defaultTheme is TRUE. Default FALSE.
summary	Adds a summary statistic, as well as a crossbar to the violin plot. Options are "mean" or "median". Default NULL.
title	Title of plot. Default NULL.
titleSize	Size of title of plot. Default 15.
combinePlot	Must be either "all", "sample", or "none". "all" will combine all plots into a single .ggplot object, while "sample" will output a list of plots separated by sample. Default "none".
plotLabels	labels to each plot. If set to "default", will use the name of the samples as the labels. If set to "none", no label will be plotted.

Value

a ggplot of the violin plot of assay data.

Examples

```
plotSCEViolinAssayData(
  inSCE = mouseBrainSubsetSCE,
  feature = "Apoe", groupBy = "sex"
)
```

---

plotSCEViolinColData *Violin plot of colData.*

Description

Visualizes values stored in the colData slot of a SingleCellExperiment object via a violin plot.

Usage

```
plotSCEViolinColData(
  inSCE,
  coldata,
  sample = NULL,
  groupBy = NULL,
  violin = TRUE,
  boxplot = TRUE,
  dots = TRUE,
  xlab = NULL,
  ylab = NULL,
  baseSize = 12,
  axisSize = NULL,
  axisLabelSize = NULL,
  dotSize = 0.5,
  transparency = 1,
  defaultTheme = TRUE,
  gridLine = FALSE,
  summary = NULL,
  summaryTextSize = 3,
  title = NULL,
  titleSize = NULL,
  combinePlot = "none",
  plotLabels = NULL
)
```

Arguments

inSCE	Input SingleCellExperiment object with saved dimension reduction components or a variable with saved results. Required.
coldata	colData value that will be plotted.
sample	Character vector. Indicates which sample each cell belongs to.
groupBy	Groupings for each numeric value. A user may input a vector equal length to the number of the samples in the SingleCellExperiment object, or can be retrieved from the colData slot. Default NULL.
violin	Boolean. If TRUE, will plot the violin plot. Default TRUE.
boxplot	Boolean. If TRUE, will plot boxplots for each violin plot. Default TRUE.
dots	Boolean. If TRUE, will plot dots for each violin plot. Default TRUE.
xlab	Character vector. Label for x-axis. Default NULL.
ylab	Character vector. Label for y-axis. Default NULL.
baseSize	The base font size for all text. Default 12. Can be overwritten by titleSize, axisSize, and axisLabelSize.
axisSize	Size of x/y-axis ticks. Default NULL.
axisLabelSize	Size of x/y-axis labels. Default NULL.
dotSize	Size of dots. Default 0.5.
transparency	Transparency of the dots, values will be 0-1. Default 1.
defaultTheme	Removes grid in plot and sets axis title size to 10 when TRUE. Default TRUE.
gridLine	Adds a horizontal grid line if TRUE. Will still be drawn even if defaultTheme is TRUE. Default FALSE.
summary	Adds a summary statistic, as well as a crossbar to the violin plot. Options are "mean" or "median". Default NULL.
summaryTextSize	The text size of the summary statistic displayed above the violin plot. Default 3.
title	Title of plot. Default NULL.
titleSize	Size of title of plot. Default 15.
combinePlot	Must be either "all", "sample", or "none". "all" will combine all plots into a single .ggplot object, while "sample" will output a list of plots separated by sample. Default "none".
plotLabels	labels to each plot. If set to "default", will use the name of the samples as the labels. If set to "none", no label will be plotted.

Value

a ggplot of the violin plot of coldata.

Examples

```
plotSCEViolinColData(
  inSCE = mouseBrainSubsetSCE,
  coldata = "age", groupBy = "sex"
)
```

plotScrubletResults *Plots for runScrublet outputs.*

Description

A wrapper function which visualizes outputs from the runScrublet function stored in the colData slot of the SingleCellExperiment object via various plots.

Usage

```
plotScrubletResults(
  inSCE,
  sample = NULL,
  shape = NULL,
  groupBy = NULL,
  combinePlot = "all",
  violin = TRUE,
  boxplot = FALSE,
  dots = TRUE,
  reducedDimName,
  xlab = NULL,
  ylab = NULL,
  dim1 = NULL,
  dim2 = NULL,
  bin = NULL,
  binLabel = NULL,
  defaultTheme = TRUE,
  dotSize = 0.5,
  summary = "median",
  summaryTextSize = 3,
  transparency = 1,
  baseSize = 15,
  titleSize = NULL,
  axisLabelSize = NULL,
  axisSize = NULL,
  legendSize = NULL,
  legendTitleSize = NULL,
  relHeights = 1,
  relWidths = c(1, 1, 1),
  plotNCols = NULL,
  plotNRows = NULL,
  labelSamples = TRUE,
  samplePerColumn = TRUE,
  sampleRelHeights = 1,
  sampleRelWidths = 1
)
```

Arguments

inSCE	Input SingleCellExperiment object with saved dimension reduction components or a variable with saved results from runScrublet. Required.
sample	Character vector. Indicates which sample each cell belongs to. Default NULL.
shape	If provided, add shapes based on the value.
groupBy	Groupings for each numeric value. A user may input a vector equal length to the number of the samples in the SingleCellExperiment object, or can be retrieved from the colData slot. Default NULL.
combinePlot	Must be either "all", "sample", or "none". "all" will combine all plots into a single .ggplot object, while "sample" will output a list of plots separated by sample. Default "all".
violin	Boolean. If TRUE, will plot the violin plot. Default TRUE.
boxplot	Boolean. If TRUE, will plot boxplots for each violin plot. Default TRUE.
dots	Boolean. If TRUE, will plot dots for each violin plot. Default TRUE.
reducedDimName	Saved dimension reduction name in the SingleCellExperiment object. Required.
xlab	Character vector. Label for x-axis. Default NULL.
ylab	Character vector. Label for y-axis. Default NULL.
dim1	1st dimension to be used for plotting. Can either be a string which specifies the name of the dimension to be plotted from reducedDims, or a numeric value which specifies the index of the dimension to be plotted. Default is NULL.
dim2	2nd dimension to be used for plotting. Can either be a string which specifies the name of the dimension to be plotted from reducedDims, or a numeric value which specifies the index of the dimension to be plotted. Default is NULL.
bin	Numeric vector. If single value, will divide the numeric values into the 'bin' groups. If more than one value, will bin numeric values using values as a cut point.
binLabel	Character vector. Labels for the bins created by the 'bin' parameter. Default NULL.
defaultTheme	Removes grid in plot and sets axis title size to 10 when TRUE. Default TRUE.
dotSize	Size of dots. Default 0.5.
summary	Adds a summary statistic, as well as a crossbar to the violin plot. Options are "mean" or "median". Default NULL.
summaryTextSize	The text size of the summary statistic displayed above the violin plot. Default 3.
transparency	Transparency of the dots, values will be 0-1. Default 1.
baseSize	The base font size for all text. Default 12. Can be overwritten by titleSize, axisSize, and axisLabelSize, legendSize, legendTitleSize.
titleSize	Size of title of plot. Default NULL.
axisLabelSize	Size of x/y-axis labels. Default NULL.
axisSize	Size of x/y-axis ticks. Default NULL.
legendSize	size of legend. Default NULL.

legendTitleSize
: size of legend title. Default NULL.

relHeights
: Relative heights of plots when combine is set.

relWidths
: Relative widths of plots when combine is set.

plotNCols
: Number of columns when plots are combined in a grid.

plotNRows
: Number of rows when plots are combined in a grid.

labelSamples
: Will label sample name in title of plot if TRUE. Default TRUE.

samplePerColumn
: If TRUE, when there are multiple samples and combining by "all", the output .ggplot will have plots from each sample on a single column. Default TRUE.

sampleRelHeights
: If there are multiple samples and combining by "all", the relative heights for each plot.

sampleRelWidths
: If there are multiple samples and combining by "all", the relative widths for each plot.

Value

list of .ggplot objects

Examples

```
data(scExample, package="singleCellTK")
## Not run:
sce <- subsetSCECols(sce, colData = "type != 'EmptyDroplet'")
sce <- getUMAP(inSCE=sce, useAssay="counts", reducedDimName="UMAP")
sce <- runScrublet(sce)
plotScrubletResults(inSCE=sce, reducedDimName="UMAP")

## End(Not run)
```

---

plotTopHVG *Plot highly variable genes*

Description

Plot highly variable genes

Usage

```
plotTopHVG(
  inSCE,
  method = c("vst", "mean.var.plot", "dispersion", "modelGeneVar"),
  hvgList = NULL,
  n = NULL,
  labelsCount = NULL
)
```

Arguments

inSCE	Input SingleCellExperiment object containing the computations.
method	Select either "vst", "mean.var.plot", "dispersion" or "modelGeneVar".
hvgList	Character vector indicating the labels of highly variable genes.
n	Specify the number of top genes to highlight in red. If hvgList parameter is not provided, this parameter can be used simply to specify the number of top genes to highlight in red.
labelsCount	Specify the number of data points/genes to label. By default, all top genes will be labeled.

Value

plot object

Examples

```
data("mouseBrainSubsetSCE", package = "singleCellTK")
mouseBrainSubsetSCE <- scranModelGeneVar(mouseBrainSubsetSCE, "logcounts")
plotTopHVG(mouseBrainSubsetSCE, method = "modelGeneVar",
           n = 1000, labelsCount = 0)
```

---

plotTSNE *Plot t-SNE plot on dimensionality reduction data run from t-SNE method.*

Description

Plot t-SNE plot on dimensionality reduction data run from t-SNE method.

Usage

```
plotTSNE(
  inSCE,
  colorBy = "No Color",
  shape = "No Shape",
  reducedDimName = "TSNE",
  runTSNE = FALSE,
  useAssay = "logcounts"
)
```

Arguments

inSCE	Input SingleCellExperiment object.
colorBy	color by condition.
shape	add shape to each distinct label.

reducedDimName	a name to store the results of the dimension reduction coordinates obtained from this method. This is stored in the SingleCellExperiment object in the reducedDims slot. Required.
runTSNE	Run t-SNE if the reducedDimName does not exist. the Default is FALSE.
useAssay	Indicate which assay to use. The default is "logcounts".

Value

A t-SNE plot

Examples

```
data("mouseBrainSubsetSCE")
plotTSNE(mouseBrainSubsetSCE, colorBy = "level1class",
         reducedDimName = "TSNE_counts")
```

---

plotUMAP *Plot UMAP results either on already run results or run first and then plot.*

Description

Plot UMAP results either on already run results or run first and then plot.

Usage

```
plotUMAP(
  inSCE,
  colorBy = "No Color",
  shape = "No Shape",
  reducedDimName = "UMAP",
  runUMAP = FALSE,
  useAssay = "logcounts"
)
```

Arguments

inSCE	Input SingleCellExperiment object with saved dimension reduction components. Required
colorBy	color by a condition(any column of the annotation data).
shape	add shapes to each condition.
reducedDimName	saved dimension reduction name in the SingleCellExperiment object. Required.
runUMAP	If the dimension reduction components are already available set this to FALSE, otherwise set to TRUE. Default is False.
useAssay	Indicate which assay to use. The default is "logcounts"

Value

a UMAP plot of the reduced dimensions.

Examples

```
data(scExample, package = "singleCellTK")
sce <- subsetSCECols(sce, colData = "type != 'EmptyDroplet'")
sce <- getUMAP(inSCE = sce, useAssay = "counts", reducedDimName = "UMAP")
plotUMAP(sce, shape = "No Shape", reducedDimName = "UMAP",
         runUMAP = TRUE, useAssay = "counts")
```

---

qcInputProcess *Create SingleCellExperiment object from command line input arguments*

Description

Create SingleCellExperiment object from command line input arguments

Usage

```
qcInputProcess(
  preproc,
  samplename,
  path,
  raw,
  fil,
  ref,
  rawFile,
  filFile,
  dataType
)
```

Arguments

preproc	Method used to preprocess the data. It's one of the path provided in –preproc argument.
samplename	The sample name of the data. It's one of the path provided in –sample argument.
path	Base path of the dataset. It's one of the path provided in –bash_path argument.
raw	The directory contains droplet matrix, gene and cell barcodes information. It's one of the path provided in –raw_data_path argument.
fil	The directory contains cell matrix, gene and cell barcodes information. It's one of the path provided in –cell_data_path argument.
ref	The name of reference used by cellranger. Only need for CellrangerV2 data.

rawFile	The full path of the RDS file or Matrix file of the raw gene count matrix. It's one of the path provided in –raw_data argument.
filFile	The full path of the RDS file or Matrix file of the cell count matrix. It's one of the path provided in –cell_data argument.
dataType	Type of the input. It can be "Both", "Droplet" or "Cell". It's one of the path provided in –genome argument.

Value

A list of SingleCellExperiment object containing the droplet or cell data or both,depending on the dataType that users provided.

`readSingleCellMatrix` *Read single cell expression matrix*

Description

Automatically detact the format of the input file and read the file.

Usage

```
readSingleCellMatrix(
  file,
  class = c("Matrix", "matrix"),
  delayedArray = TRUE,
  colIndexLocation = NULL,
  rowIndexLocation = NULL
)
```

Arguments

file	Path to input file. Supported file endings include .mtx, .txt, .csv, .tab, .tsv, .npz, and their corresponding `gzip`, `bzip2`, or `xz` compressed extensions (`*.gz`, `*.bz2`, or `*.xz`).
class	Character. Class of matrix. One of "Matrix" or "matrix". Specifying "Matrix" will convert to a sparse format which should be used for datasets with large numbers of cells. Default "Matrix".
delayedArray	Boolean. Whether to read the expression matrix as DelayedArray object or not. Default `TRUE`.
colIndexLocation	Character. For Optimus output, the path to the barcode index .npy file. Used only if `file` has .npz extension. Default `NULL`.
rowIndexLocation	Character. For Optimus output, The path to the feature (gene) index .npy file. Used only if `file` has .npz extension. Default `NULL`.

Value

A DelayedArray object or matrix.

Examples

```
mat <- readSingleCellMatrix(system.file("extdata/hgmm_1k_v3_20x20/outs/",
    "filtered_feature_bc_matrix/matrix.mtx.gz", package = "singleCellTK"))
```

---

reportCellQC *Get runCellQC .html report*

Description

A function to generate .html Rmarkdown report containing the visualizations of the runCellQC function output

Usage

```
reportCellQC(
  inSCE,
  output_file = NULL,
  output_dir = NULL,
  subTitle = NULL,
  studyDesign = NULL
)
```

Arguments

inSCE	A SingleCellExperiment object containing the filtered count matrix with the output from runCellQC function
output_file	name of the generated file. If NULL/default then the output file name will be based on the name of the Rmarkdown template.
output_dir	name of the output directory to save the rendered file. If NULL/default the file is stored to the current working directory
subTitle	subtitle of the QC HTML report. Default is NULL.
studyDesign	description of the data set and experiment design. It would be shown at the top of QC HTML report. Default is NULL.

Value

.html file

Examples

```
data(scExample, package = "singleCellTK")
sce <- subsetSCECols(sce, colData = "type != 'EmptyDroplet'")
## Not run:
sce <- runCellQC(sce)
reportCellQC(inSCE = sce)

## End(Not run)
```

`reportDiffExp` *Get runDEAnalysis .html report*

Description

A function to generate .html Rmarkdown report containing the visualizations of the `runDEAnalysis` function output

Usage

```
reportDiffExp(inSCE, study, output_file = NULL, output_dir = NULL)
```

Arguments

`inSCE`	A `SingleCellExperiment` object containing the output from `runDEAnalysis` function
`study`	The specific analysis to visualize, used as `analysisName` argument when running differential expression.
`output_file`	name of the generated file. If `NULL` then the output file name will be based on the name of the Rmarkdown template. Default `NULL`.
`output_dir`	name of the output directory to save the rendered file. If `NULL` the file is stored to the current working directory. Default `NULL`.

Value

Saves the HTML report in the specified output directory.

`reportDropletQC` *Get runDropletQC .html report*

Description

A function to generate .html Rmarkdown report containing the visualizations of the runDropletQC function output

Usage

```
reportDropletQC(
  inSCE,
  output_file = NULL,
  output_dir = NULL,
  subTitle = NULL,
  studyDesign = NULL
)
```

Arguments

`inSCE`	A SingleCellExperiment object containing the full droplet count matrix with the output from runDropletQC function
`output_file`	name of the generated file. If NULL/default then the output file name will be based on the name of the Rmarkdown template
`output_dir`	name of the output directory to save the rendered file. If NULL/default the file is stored to the current working directory
`subTitle`	subtitle of the QC HTML report. Default is NULL.
`studyDesign`	description of the data set and experiment design. It would be shown at the top of QC HTML report. Default is NULL.

Value

.html file

Examples

```
data(scExample, package = "singleCellTK")
## Not run:
sce <- runDropletQC(sce)
reportDropletQC(inSCE = sce)

## End(Not run)
```

reportFindMarker *Get findMarkerDiffExp .html report*

Description

A function to generate .html Rmarkdown report containing the visualizations of the `findMarkerDiffExp` function output

Usage

```
reportFindMarker(inSCE, output_file = NULL, output_dir = NULL)
```

Arguments

`inSCE`	A `SingleCellExperiment` object containing the output from `findMarkerDiffExp` function
`output_file`	name of the generated file. If `NULL` then the output file name will be based on the name of the Rmarkdown template. Default `NULL`.
`output_dir`	name of the output directory to save the rendered file. If `NULL` the file is stored to the current working directory. Default `NULL`.

Value

An HTML file of the report will be generated at the path specified in the arguments.

reportQCTool *Get .html report of the output of the selected QC algorithm*

Description

A function to generate .html Rmarkdown report for the specified QC algorithm output

Usage

```
reportQCTool(
  inSCE,
  algorithm = c("BarcodeRankDrops", "EmptyDrops", "QCMetrics", "Scrublet",
   "ScDblFinder", "Cxds", "Bcds", "CxdsBcdsHybrid", "DoubletFinder", "DecontX"),
  output_file = NULL,
  output_dir = NULL
)
```

Arguments

inSCE	A SingleCellExperiment object containing the count matrix (full droplets or filtered matrix, depends on the selected QC algorithm) with the output from at least one of these functions: runQCMetrics, runScrublet, runScDblFinder, runCxds, runBcds, runCxdsBcdsHybrid, runDecontX, runBarcodeRankDrops, runEmptyDrops
algorithm	Character. Specifies which QC algorithm report to generate. Available options are "BarcodeRankDrops", "EmptyDrops", "QCMetrics", "Scrublet", "ScDblFinder", "Cxds", "Bcds", "CxdsBcdsHybrid", "DoubletFinder" and "DecontX".
output_file	name of the generated file. If NULL/default then the output file name will be based on the name of the selected QC algorithm name .
output_dir	name of the output directory to save the rendered file. If NULL/default the file is stored to the current working directory

Value

.html file

Examples

```
data(scExample, package = "singleCellTK")
sce <- subsetSCECols(sce, colData = "type != 'EmptyDroplet'")
## Not run:
sce <- runDecontX(sce)
sce <- getUMAP(sce)
reportQCTool(inSCE = sce, algorithm = "DecontX")

## End(Not run)
```

---

retrieveSCEIndex *Retrieve cell/feature index by giving identifiers saved in col/rowData*

Description

Originally written in retrieveFeatureIndex. Modified for also retrieving cell indices and only working for SingleCellExperiment object. This will return indices of features among the rowData/colData. Partial matching (i.e. grepping) can be used.

Usage

```
retrieveSCEIndex(
  inSCE,
  IDs,
  axis,
  by = NULL,
  exactMatch = TRUE,
  firstMatch = TRUE
)
```

Arguments

inSCE	Input SingleCellExperiment object. Required
IDs	Character vector of identifiers for features or cells to find in rowData or colData of inSCE
axis	A character scalar to specify whether to search for features or cells. Use "row", "feature" or "gene" for features; "col" or "cell" for cells.
by	Character. In which column to search for features/cells in rowData/colData. Default NULL for search the rownames/colnames
exactMatch	A logical scalar. Whether to only identify exact matches or to identify partial matches using grep. Default TRUE
firstMatch	A logical scalar. Whether to only identify the first matches or to return all plausible matches. Default TRUE

Value

A unique, non-NA numeric vector of indices for the matching features/cells in inSCE.

Author(s)

Yusuke Koga, Joshua Campbell

Examples

```
data(scExample, package = "singleCellTK")
retrieveSCEIndex(inSCE = sce, IDs = "ENSG00000205542",
 axis = "row")
```

---

runANOVA *Perform differential expression analysis on SCE with ANOVA*

Description

Condition specification allows two methods: 1. Index level selection. Arguments index1 and index2 will be used. 2. Annotation level selection. Arguments class, classGroup1 and classGroup2 will be used.

Usage

```
runANOVA(
  inSCE,
  useAssay = "logcounts",
  index1 = NULL,
  index2 = NULL,
  class = NULL,
  classGroup1 = NULL,
  classGroup2 = NULL,
```

```
  analysisName,
  groupName1,
  groupName2,
  covariates = NULL,
  onlyPos = FALSE,
  log2fcThreshold = 0.25,
  fdrThreshold = 0.05,
  overwrite = FALSE
)
```

Arguments

inSCE	SingleCellExperiment inherited object.
useAssay	character. A string specifying which assay to use for ANOVA. Default "logcounts".
index1	Any type of indices that can subset a SingleCellExperiment inherited object by cells. Specifies which cells are of interests. Default NULL.
index2	Any type of indices that can subset a SingleCellExperiment inherited object by cells. specifies the control group against those specified by index1. If NULL when using index specification, index1 cells will be compared with all other cells. Default NULL.
class	A vector/factor with ncol(inSCE) elements, or a character scalar that specifies a column name of colData(inSCE). Default NULL.
classGroup1	a vector specifying which "levels" given in class are of interests. Default NULL.
classGroup2	a vector specifying which "levels" given in class is the control group against those specified by classGroup1. If NULL when using annotation specification, classGroup1 cells will be compared with all other cells.
analysisName	A character scalar naming the DEG analysis. Required
groupName1	A character scalar naming the group of interests. Required.
groupName2	A character scalar naming the control group. Required.
covariates	A character vector of additional covariates to use when building the model. All covariates must exist in names(colData(inSCE)). Default NULL.
onlyPos	Whether to only output DEG with positive log2_FC value. Default FALSE.
log2fcThreshold	Only out put DEGs with the absolute values of log2FC greater than this value. Default 0.25
fdrThreshold	Only out put DEGs with FDR value less than this value. Default 0.05
overwrite	A logical scalar. Whether to overwrite result if exists. Default FALSE.

Details

NOTE that ANOVA method does not produce Log2FC value, but P-value and FDR only.

Value

The input SingleCellExperiment object with `metadata(inSCE)$diffExp` updated with the results: a list named by `analysisName`, with `$groupNames` containing the naming of the two conditions, `$useAssay` storing the assay name that was used for calculation, `$select` storing the cell selection indices (logical) for each condition, `$result` storing a `data.frame` of the DEGs summary, and `$method` storing `"ANOVA"`.

Examples

```
data(scExample, package = "singleCellTK")
sce <- subsetSCECols(sce, colData = "type != 'EmptyDroplet'")
sce <- scaterlogNormCounts(sce, assayName = "logcounts")
sce <- runANOVA(inSCE = sce, groupName1 = "Sample1",
 groupName2 = "Sample2", index1 = seq(20), index2 = seq(21,40),
 analysisName = "ANOVA", fdrThreshold = NULL)
```

---

runBarcodeRankDrops *Identify empty droplets using barcodeRanks.*

Description

Run barcodeRanks on a count matrix provided in a SingleCellExperiment object. Distinguish between droplets containing cells and ambient RNA in a droplet-based single-cell RNA sequencing experiment.

Usage

```
runBarcodeRankDrops(
  inSCE,
  sample = NULL,
  useAssay = "counts",
  lower = 100,
  fitBounds = NULL,
  df = 20
)
```

Arguments

`inSCE`	A SingleCellExperiment object. Must contain a raw counts matrix before empty droplets have been removed.
`sample`	Character vector. Indicates which sample each cell belongs to emptyDrops will be run on cells from each sample separately. If NULL, then all cells will be processed together. Default `NULL`.
`useAssay`	A string specifying which assay in the SCE to use.
`lower`	See emptyDrops for more information. Default `100`.
`fitBounds`	See emptyDrops for more information. Default `NULL`.
`df`	See emptyDrops for more information. Default `20`.

Value

A SingleCellExperiment object with the barcodeRanks output table appended to the colData slot. The columns include *dropletUtils_BarcodeRank_Knee* and *dropletUtils_BarcodeRank_Knee* Please refer to the documentation of barcodeRanks for details.

Examples

```
# The following unfiltered PBMC_1k_v3 data were downloaded from
# https://support.10xgenomics.com/single-cell-gene-expression/datasets/3.0.0
# /pbmc_1k_v3
# Only the top 10 cells with most counts and the last 10 cells with non-zero
# counts are included in this example.
# This example only serves as an proof of concept and a tutoriol on how to
# run the function. The results should not be
# used for drawing scientific conclusions.
data(scExample, package = "singleCellTK")
sce <- runBarcodeRankDrops(inSCE = sce)
```

---

runBBKNN *Apply BBKNN batch effect correction method to SingleCellExperiment object*

Description

BBKNN, an extremely fast graph-based data integration algorithm. It modifies the neighbourhood construction step to produce a graph that is balanced across all batches of the data.

Usage

```
runBBKNN(
  inSCE,
  useAssay = "logcounts",
  batch = "batch",
  reducedDimName = "BBKNN",
  nComponents = 50L
)
```

Arguments

inSCE	SingleCellExperiment inherited object. Required.
useAssay	A single character indicating the name of the assay requiring batch correction. Default "logcounts".
batch	A single character indicating a field in colData that annotates the batches. Default "batch".
reducedDimName	A single character. The name for the corrected low-dimensional representation. Will be saved to reducedDim(inSCE). Default "BBKNN".

nComponents An integer. Number of principle components or the dimensionality, adopted in the pre-PCA-computation step, the BBKNN step (for how many PCs the algorithm takes into account), and the final UMAP combination step where the value represent the dimensionality of the updated reducedDim. Default `50L`.

Value

The input SingleCellExperiment object with `reducedDim(inSCE,reducedDimName)` updated.

References

Krzysztof Polanski et al., 2020

Examples

```
## Not run:
data('sceBatches', package = 'singleCellTK')
logcounts(sceBatches) <- log(counts(sceBatches) + 1)
sceBatches.small <- sample(sceBatches, 20)
sceCorr <- runBBKNN(sceBatches.small, useAssay = "logcounts",
                    nComponents = 10)

## End(Not run)
```

---

runBcds *Find doublets/multiplets using bcds.*

Description

A wrapper function for bcds. Annotate doublets/multiplets using a binary classification approach to discriminate artificial doublets from original data. Generate a doublet score for each cell. Infer doublets if `estNdbl` is `TRUE`.

Usage

```
runBcds(
  inSCE,
  sample = NULL,
  seed = 12345,
  ntop = 500,
  srat = 1,
  verb = FALSE,
  retRes = FALSE,
  nmax = "tune",
  varImp = FALSE,
  estNdbl = FALSE,
  useAssay = "counts"
)
```

Arguments

inSCE	A SingleCellExperiment object. Needs counts in assays slot.
sample	Character vector. Indicates which sample each cell belongs to. bcds will be run on cells from each sample separately. If NULL, then all cells will be processed together. Default NULL.
seed	Seed for the random number generator. Default 12345.
ntop	See bcds for more information. Default 500.
srat	See bcds for more information. Default 1.
verb	See bcds for more information. Default FALSE.
retRes	See bcds for more information. Default FALSE.
nmax	See bcds for more information. Default "tune".
varImp	See bcds for more information. Default FALSE.
estNdbl	See bcds for more information. Default FALSE.
useAssay	A string specifying which assay in the SCE to use.

Value

A SingleCellExperiment object with bcds output appended to the colData slot. The columns include *bcds_score* and optionally *bcds_call*. Please refer to the documentation of bcds for details.

Examples

```
data(scExample, package = "singleCellTK")
sce <- subsetSCECols(sce, colData = "type != 'EmptyDroplet'")
sce <- runBcds(sce)
```

---

runCellQC *Perform comprehensive single cell QC*

Description

A wrapper function to run several QC algorithms on a SingleCellExperiment object containing cells after empty droplets have been removed.

Usage

```
runCellQC(
  inSCE,
  algorithms = c("QCMetrics", "scDblFinder", "cxds", "bcds", "cxds_bcds_hybrid",
    "decontX"),
  sample = NULL,
  collectionName = NULL,
  geneSetList = NULL,
  geneSetListLocation = "rownames",
```

```
  geneSetCollection = NULL,
  useAssay = "counts",
  seed = 12345,
  paramsList = NULL
)
```

Arguments

inSCE	A SingleCellExperiment object.
algorithms	Character vector. Specify which QC algorithms to run. Available options are "QCMetrics", "scrublet", "doubletFinder", "scDblFinder", "cxds", "bcds", "cxds_bcds_hybrid", and "decontX".
sample	Character vector. Indicates which sample each cell belongs to. Algorithms will be run on cells from each sample separately.
collectionName	Character. Name of a GeneSetCollection obtained by using one of the import-GeneSet* functions. Default NULL.
geneSetList	See runPerCellQC. Default NULL.
geneSetListLocation	See runPerCellQC. Default NULL.
geneSetCollection	See runPerCellQC. Default NULL.
useAssay	A string specifying which assay contains the count matrix for cells.
seed	Seed for the random number generator. Default 12345.
paramsList	A list containing parameters for QC functions. Default NULL.

Value

SingleCellExperiment object containing the outputs of the specified algorithms in the colData of inSCE.

Examples

```
data(scExample, package = "singleCellTK")
sce <- subsetSCECols(sce, colData = "type != 'EmptyDroplet'")
## Not run:
sce <- runCellQC(sce)

## End(Not run)
```

runComBatSeq *Apply ComBat-Seq batch effect correction method to SingleCellExperiment object*

Description

The ComBat-Seq batch adjustment approach assumes that batch effects represent non-biological but systematic shifts in the mean or variability of genomic features for all samples within a processing batch. It uses either parametric or non-parametric empirical Bayes frameworks for adjusting data for batch effects.

Usage

```
runComBatSeq(
  inSCE,
  useAssay = "counts",
  batch = "batch",
  covariates = NULL,
  bioCond = NULL,
  useSVA = FALSE,
  assayName = "ComBatSeq",
  shrink = FALSE,
  shrinkDisp = FALSE,
  nGene = NULL
)
```

Arguments

inSCE	SingleCellExperiment inherited object. Required.
useAssay	A single character indicating the name of the assay requiring batch correction. Default "counts".
batch	A single character indicating a field in colData that annotates the batches. Default "batch".
covariates	A character vector indicating the fields in colData that annotates other covariates, such as the cell types. Default NULL.
bioCond	A single character indicating a field in colData that annotates the biological conditions. Default NULL.
useSVA	A logical scalar. Whether to estimate surrogate variables and use them as an empirical control. Default FALSE.
assayName	A single characeter. The name for the corrected assay. Will be saved to assay. Default "ComBat".
shrink	A logical scalar. Whether to apply shrinkage on parameter estimation. Default FALSE.
shrinkDisp	A logical scalar. Whether to apply shrinkage on dispersion. Default FALSE.
nGene	An integer. Number of random genes to use in empirical Bayes estimation, only useful when shrink is set to TRUE. Default NULL.

Details

For the parameters `covariates` and `useSVA`, when the cell type information is known, it is recommended to specify the cell type annotation to the argument `covariates`; if the cell types are unknown but expected to be balanced, it is recommended to run with default settings, yet informative covariates could still be useful. If the cell types are unknown and are expected to be unbalanced, it is recommended to set `useSVA` to `TRUE`.

Value

The input SingleCellExperiment object with `assay(inSCE,assayName)` updated.

Examples

```
data('sceBatches', package = 'singleCellTK')
sceBatches <- sample(sceBatches, 40)
# Cell type known
sceBatches <- runComBatSeq(sceBatches, "counts", "batch",
                           covariates = "cell_type",
                           assayName = "ComBat_cell_seq")
# Cell type unknown but balanced
#sceBatches <- runComBatSeq(sceBatches, "counts", "batch",
#                           assayName = "ComBat_seq")
# Cell type unknown and unbalanced
#sceBatches <- runComBatSeq(sceBatches, "counts", "batch",
#                           useSVA = TRUE,
#                           assayName = "ComBat_sva_seq")
```

---

runCxds *Find doublets/multiplets using cxds.*

Description

A wrapper function for cxds. Annotate doublets/multiplets using co-expression based approach. Generate a doublet score for each cell. Infer doublets if `estNdbl` is `TRUE`.

Usage

```
runCxds(
  inSCE,
  sample = NULL,
  seed = 12345,
  ntop = 500,
  binThresh = 0,
  verb = FALSE,
  retRes = FALSE,
  estNdbl = FALSE,
  useAssay = "counts"
)
```

Arguments

inSCE	A SingleCellExperiment object. Needs counts in assays slot.
sample	Character vector. Indicates which sample each cell belongs to. cxds will be run on cells from each sample separately. If NULL, then all cells will be processed together. Default NULL.
seed	Seed for the random number generator. Default 12345.
ntop	See cxds for more information. Default 500.
binThresh	See cxds for more information. Default 0.
verb	See cxds for more information. Default FALSE.
retRes	See cxds for more information. Default FALSE.
estNdbl	See cxds for more information. Default FALSE.
useAssay	A string specifying which assay in the SCE to use.

Value

A SingleCellExperiment object with cxds output appended to the colData slot. The columns include *cxds_score* and optionally *cxds_call*. Please refer to the documentation of cxds for details.

Examples

```
data(scExample, package = "singleCellTK")
sce <- subsetSCECols(sce, colData = "type != 'EmptyDroplet'")
sce <- runCxds(sce)
```

---

runCxdsBcdsHybrid *Find doublets/multiplets using cxds_bcds_hybrid.*

Description

A wrapper function for cxds_bcds_hybrid. Annotate doublets/multiplets using a binary classification approach to discriminate artificial doublets from original data. Generate a doublet score for each cell. Infer doublets if estNdbl is TRUE.

Usage

```
runCxdsBcdsHybrid(
  inSCE,
  sample = NULL,
  seed = 12345,
  nTop = 500,
  cxdsArgs = list(),
  bcdsArgs = list(),
  verb = FALSE,
  estNdbl = FALSE,
  force = FALSE,
  useAssay = "counts"
)
```

Arguments

inSCE	A SingleCellExperiment object. Needs counts in assays slot.
sample	Character vector. Indicates which sample each cell belongs to. cxds_bcds_hybrid will be run on cells from each sample separately. If NULL, then all cells will be processed together. Default NULL.
seed	Seed for the random number generator. Default 12345.
nTop	The number of top varialbe genes to consider. Used in both csds and bcds. Default 500.
cxdsArgs	See cxds_bcds_hybrid for more information. Default NULL.
bcdsArgs	See cxds_bcds_hybrid for more information. Default NULL.
verb	See cxds_bcds_hybrid for more information. Default FALSE.
estNdbl	See cxds_bcds_hybrid for more information. Default FALSE.
force	See cxds_bcds_hybrid for more information. Default FALSE.
useAssay	A string specifying which assay in the SCE to use.

Value

A SingleCellExperiment object with cxds_bcds_hybrid output appended to the colData slot. The columns include *hybrid_score* and optionally *hybrid_call*. Please refer to the documentation of cxds_bcds_hybrid for details.

Examples

```
data(scExample, package = "singleCellTK")
sce <- subsetSCECols(sce, colData = "type != 'EmptyDroplet'")
sce <- runCxdsBcdsHybrid(sce)
```

---

runDEAnalysis *Perform differential expression analysis on SCE with specified method Method supported: 'MAST', 'DESeq2', 'Limma', 'ANOVA'*

Description

Perform differential expression analysis on SCE with specified method Method supported: 'MAST', 'DESeq2', 'Limma', 'ANOVA'

Usage

```
runDEAnalysis(method = c("MAST", "DESeq2", "Limma", "ANOVA", "wilcox"), ...)
```

Arguments

method	A single character for specific method. Choose from "MAST", "DESeq2", "Limma", "ANOVA". Default "MAST".
...	Other arguments passed to specific functions. Refer to runMAST, runDESeq2, runLimmaDE, runANOVA

Value

Input SCE object with `metadata(inSCE)` updated with name `"diffExp"` as a `list` object. Detail refers to the four child functions.

Examples

```
data(scExample, package = "singleCellTK")
sce <- subsetSCECols(sce, colData = "type != 'EmptyDroplet'")
sce <- scaterlogNormCounts(sce, "logcounts")
sce <- runDEAnalysis(inSCE = sce, groupName1 = "Sample1", method = "wilcox",
 groupName2 = "Sample2", index1 = seq(20), index2 = seq(21,40),
 analysisName = "Limma")
```

`runDecontX` *Detecting contamination with DecontX.*

Description

A wrapper function for decontX. Identify potential contamination from experimental factors such as ambient RNA.

Usage

```
runDecontX(
  inSCE,
  sample = NULL,
  useAssay = "counts",
  z = NULL,
  maxIter = 500,
  delta = c(10, 10),
  estimateDelta = TRUE,
  convergence = 0.001,
  iterLogLik = 10,
  varGenes = 5000,
  dbscanEps = 1,
  seed = 12345,
  logfile = NULL,
  verbose = TRUE
)
```

Arguments

`inSCE`	A SingleCellExperiment object.
`sample`	A single character specifying a name that can be found in `colData(inSCE)` to directly use the cell annotation; or a character vector with as many elements as cells to indicates which sample each cell belongs to. Default NULL. decontX will be run on cells from each sample separately.

useAssay	A string specifying which assay in the SCE to use. Default 'counts'.
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 dimenions and the 'dbscan' function from the 'dbscan' package will be used to identify clusters of broad cell types. Default NULL.
maxIter	Integer. Maximum iterations of the EM algorithm. Default 500.
delta	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.
iterLogLik	Integer. Calculate log likelihood every `iterLogLik` iteration. Default 10.
varGenes	Integer. The number of variable genes to use in dimensionality reduction before clustering. Variability is calcualted 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

A SingleCellExperiment object with 'decontX_Contamination' and 'decontX_Clusters' added to the colData slot. Additionally, the decontaminated counts will be added as an assay called 'decontXCounts'.

Examples

```
data(scExample, package = "singleCellTK")
sce <- subsetSCECols(sce, colData = "type != 'EmptyDroplet'")
sce <- runDecontX(sce[,sample(ncol(sce),20)])
```

runDESeq2 *Perform differential expression analysis on SCE with DESeq2.*

Description

Condition specification allows two methods: 1. Index level selection. Arguments `index1` and `index2` will be used. 2. Annotation level selection. Arguments `class`, `classGroup1` and `classGroup2` will be used.

Usage

```
runDESeq2(
  inSCE,
  useAssay = "counts",
  index1 = NULL,
  index2 = NULL,
  class = NULL,
  classGroup1 = NULL,
  classGroup2 = NULL,
  analysisName,
  groupName1,
  groupName2,
  covariates = NULL,
  fullReduced = TRUE,
  onlyPos = FALSE,
  log2fcThreshold = NULL,
  fdrThreshold = 1,
  overwrite = FALSE
)
```

Arguments

`inSCE`	SingleCellExperiment inherited object.
`useAssay`	character. A string specifying which assay to use for the DESeq2 regression. The assay should be a raw count assay. Default `"counts"`.
`index1`	Any type of indices that can subset a SingleCellExperiment inherited object by cells. Specifies which cells are of interests. Default `NULL`.
`index2`	Any type of indices that can subset a SingleCellExperiment inherited object by cells. specifies the control group against those specified by `index1`. If `NULL` when using index specification, `index1` cells will be compared with all other cells. Default `NULL`.
`class`	A vector/factor with `ncol(inSCE)` elements, or a character scalar that specifies a column name of `colData(inSCE)`. Default `NULL`.
`classGroup1`	a vector specifying which "levels" given in `class` are of interests. Default `NULL`.

classGroup2	a vector specifying which "levels" given in class is the control group against those specified by classGroup1. If NULL when using annotation specification, classGroup1 cells will be compared with all other cells.
analysisName	A character scalar naming the DEG analysis. Required
groupName1	A character scalar naming the group of interests. Required.
groupName2	A character scalar naming the control group. Required.
covariates	A character vector of additional covariates to use when building the model. All covariates must exist in names(colData(inSCE)). Default NULL.
fullReduced	Whether to apply LRT (Likelihood ratio test) with a 'full' model. Default TRUE.
onlyPos	Whether to only output DEG with positive log2_FC value. Default FALSE.
log2fcThreshold	Only out put DEGs with the absolute values of log2FC greater than this value. Default 0.25
fdrThreshold	Only out put DEGs with FDR value less than this value. Default 0.05
overwrite	A logical scalar. Whether to overwrite result if exists. Default FALSE.

Value

The input SingleCellExperiment object with metadata(inSCE)$DESeq2 updated with the results: a list named by analysisName, with $groupNames containing the naming of the two conditions, $useAssay storing the assay name that was used for calculation, $select storing the cell selection indices (logical) for each condition, $result storing a data.frame of the DEGs summary, and $method storing "DESeq2".

Examples

```
data(scExample, package = "singleCellTK")
sce <- subsetSCECols(sce, colData = "type != 'EmptyDroplet'")
sce <- runDESeq2(inSCE = sce, groupName1 = "Sample1",
 groupName2 = "Sample2", index1 = seq(5), index2 = seq(6,10),
 analysisName = "DESeq2")
```

---

runDimReduce *Generic Wrapper function for running dimensionality reduction*

Description

Generic Wrapper function for running dimensionality reduction

Usage

```
runDimReduce(
  inSCE,
  method = c("scaterPCA", "seuratPCA", "seuratICA", "rTSNE", "seuratTSNE",
    "scaterUMAP", "seuratUMAP"),
  useAssay = NULL,
  useReducedDim = NULL,
  useAltExp = NULL,
  reducedDimName,
  nComponents = 20,
  ...
)
```

Arguments

inSCE	Input SingleCellExperiment object.
method	One from "scaterPCA", "seuratPCA", "seuratICA", "rTSNE", "seuratTSNE", "scaterUMAP" and "seuratUMAP".
useAssay	Assay to use for computation. If useAltExp is specified, useAssay has to exist in assays(altExp(inSCE,useAltExp)). Default "counts".
useReducedDim	The low dimension representation to use for embedding computation. Default NULL.
useAltExp	The subset to use for computation, usually for the selected variable features. Default NULL.
reducedDimName	The name of the result matrix. Required.
nComponents	Specify the number of dimensions to compute with the selected method in case of PCA/ICA and the number of components to use in the case of TSNE/UMAP methods.
...	The other arguments for running a specific algorithm. Please refer to the one you use.

Details

Wrapper function to run one of the available dimensionality reduction algorithms integrated within SCTK from scaterPCA, seuratPCA, seuratICA, getTSNE, seuratRunTSNE, getUMAP and seuratRunUMAP. Users can use an assay by specifying useAssay, use the assay in an altExp by specifying both useAltExp and useAssay, or use a low-dimensionality representation by specifying useReducedDim.

Value

The input SingleCellExperiment object with reducedDim updated with the result.

Examples

```
data(scExample, package = "singleCellTK")
sce <- subsetSCECols(sce, colData = "type != 'EmptyDroplet'")
sce <- runNormalization(sce, useAssay = "counts",
```

```
                        outAssayName = "logcounts_scaled",
                        normalizationMethod = "logNormCounts",
                        scale = TRUE)
sce <- runDimReduce(inSCE = sce, method = "scaterPCA",
                    useAssay = "logcounts_scaled",
                    reducedDimName = "PCA")
```

`runDoubletFinder` *Generates a doublet score for each cell via doubletFinder*

Description

Uses doubletFinder to determine cells within the dataset suspected to be doublets.

Usage

```
runDoubletFinder(
  inSCE,
  useAssay = "counts",
  sample = NULL,
  seed = 12345,
  seuratNfeatures = 2000,
  seuratPcs = seq(15),
  seuratRes = 1.5,
  formationRate = 0.075,
  nCores = NULL,
  verbose = FALSE
)
```

Arguments

`inSCE`	Input SingleCellExperiment object. Must contain a counts matrix
`useAssay`	Indicate which assay to use. Default "counts".
`sample`	Numeric vector. Each cell will be assigned a sample number.
`seed`	Seed for the random number generator. Default 12345.
`seuratNfeatures`	Integer. Number of highly variable genes to use. Default 2000.
`seuratPcs`	Numeric vector. The PCs used in seurat function to determine number of clusters. Default 1:15.
`seuratRes`	Numeric vector. The resolution parameter used in seurat, which adjusts the number of clusters determined via the algorithm. Default 1.5.
`formationRate`	Doublet formation rate used within algorithm. Default 0.075.
`nCores`	Number of cores used for running the function.
`verbose`	Boolean. Wheter to print messages from Seurat and DoubletFinder. Default FALSE.

Value

SingleCellExperiment object containing the 'doublet_finder_doublet_score'.

Examples

```
data(scExample, package = "singleCellTK")
sce <- subsetSCECols(sce, colData = "type != 'EmptyDroplet'")
sce <- runDoubletFinder(sce)
```

---

runDropletQC *Perform comprehensive droplet QC*

Description

A wrapper function to run several QC algorithms for determining empty droplets in single cell RNA-seq data

Usage

```
runDropletQC(
  inSCE,
  algorithms = c("QCMetrics", "emptyDrops", "barcodeRanks"),
  sample = NULL,
  useAssay = "counts",
  paramsList = NULL
)
```

Arguments

inSCE	A SingleCellExperiment object containing the full droplet count matrix
algorithms	Character vector. Specify which QC algorithms to run. Available options are "emptyDrops" and "barcodeRanks".
sample	Character vector. Indicates which sample each cell belongs to. Algorithms will be run on cells from each sample separately.
useAssay	A string specifying which assay contains the count matrix for droplets.
paramsList	A list containing parameters for QC functions. Default NULL.

Value

SingleCellExperiment object containing the outputs of the specified algorithms in the colData of inSCE.

Examples

```
data(scExample, package = "singleCellTK")
## Not run:
sce <- runDropletQC(sce)

## End(Not run)
```

`runEmptyDrops` *Identify empty droplets using emptyDrops.*

Description

Run emptyDrops on the count matrix in the provided SingleCellExperiment object. Distinguish between droplets containing cells and ambient RNA in a droplet-based single-cell RNA sequencing experiment.

Usage

```
runEmptyDrops(
  inSCE,
  sample = NULL,
  useAssay = "counts",
  lower = 100,
  niters = 10000,
  testAmbient = FALSE,
  ignore = NULL,
  alpha = NULL,
  retain = NULL,
  barcodeArgs = list(),
  BPPARAM = BiocParallel::SerialParam()
)
```

Arguments

`inSCE`	Input SingleCellExperiment object. Must contain a raw counts matrix before empty droplets have been removed.
`sample`	Character vector. Indicates which sample each cell belongs to. emptyDrops will be run on cells from each sample separately. If NULL, then all cells will be processed together. Default NULL.
`useAssay`	A string specifying which assay in the SCE to use.
`lower`	See emptyDrops for more information.
`niters`	See emptyDrops for more information.
`testAmbient`	See emptyDrops for more information.
`ignore`	See emptyDrops for more information.
`alpha`	See emptyDrops for more information.

retain	See emptyDrops for more information.
barcodeArgs	See emptyDrops for more information.
BPPARAM	See emptyDrops for more information.

Value

A SingleCellExperiment object with the emptyDrops output table appended to the colData slot. The columns include *emptyDrops_total*, *emptyDrops_logprob*, *emptyDrops_pvalue*, *emptyDrops_limited*, *emptyDrops_fdr*. Please refer to the documentation of emptyDrops for details.

Examples

```
# The following unfiltered PBMC_1k_v3 data were downloaded from
# https://support.10xgenomics.com/single-cell-gene-expression/datasets/3.0.0
# /pbmc_1k_v3
# Only the top 10 cells with most counts and the last 10 cells with non-zero
# counts are included in this example.
# This example only serves as an proof of concept and a tutorial on how to
# run the function. The results should not be
# used for drawing scientific conclusions.
data(scExample, package = "singleCellTK")
sce <- runEmptyDrops(inSCE = sce)
```

---

runFastMNN *Apply a fast version of the mutual nearest neighbors (MNN) batch effect correction method to SingleCellExperiment object*

Description

fastMNN is a variant of the classic MNN method, modified for speed and more robust performance. For introduction of MNN, see runMNNCorrect.

Usage

```
runFastMNN(
  inSCE,
  useAssay = "logcounts",
  reducedDimName = "fastMNN",
  batch = "batch",
  pcInput = FALSE
)
```

Arguments

inSCE	inherited object. Required.
useAssay	A single character indicating the name of the assay requiring batch correction. Default "logcounts". Alternatively, see pcInput parameter.

reducedDimName	A single character. The name for the corrected low-dimensional representation. Will be saved to reducedDim(inSCE). Default "fastMNN".
batch	A single character indicating a field in colData that annotates the batches. Default "batch".
pcInput	A logical scalar. Whether to use a low-dimension matrix for batch effect correction. If TRUE, useAssay will be searched from reducedDimNames(inSCE). Default FALSE.

Value

The input SingleCellExperiment object with reducedDim(inSCE,reducedDimName) updated.

References

Lun ATL, et al., 2016

Examples

```
data('sceBatches', package = 'singleCellTK')
logcounts(sceBatches) <- log(counts(sceBatches) + 1)
sceCorr <- runFastMNN(sceBatches, useAssay = 'logcounts', pcInput = FALSE)
```

---

runFeatureSelection *Wrapper function to run all of the feature selection methods integrated within the singleCellTK package including three methods from Seurat ('vst', 'mean.var.plot' or 'dispersion') and the Scran 'modelGeneVar' method.*

Description

Wrapper function to run all of the feature selection methods integrated within the singleCellTK package including three methods from Seurat ('vst', 'mean.var.plot' or 'dispersion') and the Scran 'modelGeneVar' method.

Usage

```
runFeatureSelection(
  inSCE,
  useAssay,
  hvgMethod = c("vst", "mean.var.plot", "dispersion", "modelGeneVar")
)
```

Arguments

inSCE	Input SingleCellExperiment object.
useAssay	Specify the name of the assay that should be used. A normalized assay is recommended for use with this function.
hvgMethod	Specify the method to use for variable gene selection. Options include "vst", "mean.var.plot" or "dispersion" from Seurat and "modelGeneVar" from Scran.

Value

A SingleCellExperiment object that contains the computed statistics in the rowData slot of the output object. This function does not return the names of the variable features but only computes the statistics that are stored in the rowData slot of the. To get the names of the variable features getTopHVG function should be used after computing these statistics.

Examples

```
data("mouseBrainSubsetSCE", package = "singleCellTK")
mouseBrainSubsetSCE <- runFeatureSelection(mouseBrainSubsetSCE,
                                           "logcounts",
                                           "modelGeneVar")
```

---

runGSVA *Run GSVA analysis on a SingleCellExperiment object*

Description

Run GSVA analysis on a SingleCellExperiment object

Usage

```
runGSVA(
  inSCE,
  useAssay = "logcounts",
  resultNamePrefix = NULL,
  geneSetCollectionName,
  ...
)
```

Arguments

inSCE	Input SingleCellExperiment object.
useAssay	Indicate which assay to use. The default is "logcounts"

resultNamePrefix
: Character. Prefix to the name the VAM results which will be stored in the reducedDim slot of inSCE. The names of the output matrices will be resultNamePrefix_Distance and resultNamePrefix_CDF. If this parameter is set to NULL, then "VAM_geneSetCollectionName_" will be used. Default NULL.

geneSetCollectionName
: Character. The name of the gene set collection to use. parameter.

...
: Parameters to pass to gsva()

Value

A SingleCellExperiment object with pathway activity scores from GSVA stored in reducedDim as GSVA__NameOfTheGeneset_Scores.

Examples

```
data(scExample, package = "singleCellTK")
sce <- subsetSCECols(sce, colData = "type != 'EmptyDroplet'")
sce <- scaterlogNormCounts(sce, assayName = "logcounts")
gs1 <- rownames(sce)[seq(10)]
gs2 <- rownames(sce)[seq(11,20)]
gs <- list("geneset1" = gs1, "geneset2" = gs2)

sce <- importGeneSetsFromList(inSCE = sce,geneSetList = gs,
                                            by = "rownames")
sce <- runGSVA(inSCE = sce, geneSetCollectionName = "GeneSetCollection", useAssay = "logcounts")
```

---

runKMeans *Get clustering with KMeans*

Description

Perform KMeans clustering on a SingleCellExperiment object, with kmeans.

Usage

```
runKMeans(
  inSCE,
  useReducedDim = "PCA",
  clusterName = "KMeans_cluster",
  nCenters,
  nIter = 10,
  nStart = 1,
  seed = 12345,
  algorithm = c("Hartigan-Wong", "Lloyd", "MacQueen")
)
```

Arguments

inSCE	A SingleCellExperiment object.
useReducedDim	A single character, specifying which low-dimension representation to perform the clustering algorithm on. Default "PCA".
clusterName	A single character, specifying the name to store the cluster label in colData. Default "scranSNN_cluster".
nCenters	An integer, the number of centroids (clusters).
nIter	An integer, the maximum number of iterations allowed. Default 10.
nStart	An integer, the number of random sets to choose. Default 1.
seed	An integer. The seed for the random number generator. Default 12345.
algorithm	A single character. Choose from "Hartigan-Wong", "Lloyd", "MacQueen". May be abbreviated. Default "Hartigan-Wong".

Value

The input SingleCellExperiment object with factor cluster labeling updated in colData(inSCE)[[clusterName]].

Examples

```
data("mouseBrainSubsetSCE")
mouseBrainSubsetSCE <- runKMeans(mouseBrainSubsetSCE,
                                 useReducedDim = "PCA_logcounts",
                                 nCenters = 2)
```

---

runLimmaBC *Apply Limma's batch effect correction method to SingleCellExperiment object*

Description

Limma's batch effect removal function fits a linear model to the data, then removes the component due to the batch effects.

Usage

```
runLimmaBC(inSCE, useAssay = "logcounts", assayName = "LIMMA", batch = "batch")
```

Arguments

inSCE	SingleCellExperiment inherited object. Required.
useAssay	A single character indicating the name of the assay requiring batch correction. Default "logcounts".
assayName	A single characeter. The name for the corrected assay. Will be saved to assay. Default "LIMMA".
batch	A single character indicating a field in colData that annotates the batches. Default "batch".

Value

The input SingleCellExperiment object with `assay(inSCE,assayName)` updated.

References

Gordon K Smyth, et al., 2003

Examples

```
data('sceBatches', package = 'singleCellTK')
logcounts(sceBatches) <- log(counts(sceBatches) + 1)
sceCorr <- runLimmaBC(sceBatches)
```

`runLimmaDE` *Perform differential expression analysis on SCE with Limma.*

Description

Condition specification allows two methods: 1. Index level selection. Arguments `index1` and `index2` will be used. 2. Annotation level selection. Arguments `class`, `classGroup1` and `classGroup2` will be used.

Usage

```
runLimmaDE(
  inSCE,
  useAssay = "logcounts",
  index1 = NULL,
  index2 = NULL,
  class = NULL,
  classGroup1 = NULL,
  classGroup2 = NULL,
  analysisName,
  groupName1,
  groupName2,
  covariates = NULL,
  onlyPos = FALSE,
  log2fcThreshold = 0.25,
  fdrThreshold = 0.05,
  overwrite = FALSE
)
```

Arguments

`inSCE`	SingleCellExperiment inherited object.
`useAssay`	character. A string specifying which assay to use for the Limma regression. The assay should be a log-transformed normalized assay. Default `"logcounts"`.

index1	Any type of indices that can subset a SingleCellExperiment inherited object by cells. Specifies which cells are of interests. Default NULL.
index2	Any type of indices that can subset a SingleCellExperiment inherited object by cells. specifies the control group against those specified by index1. If NULL when using index specification, index1 cells will be compared with all other cells. Default NULL.
class	A vector/factor with ncol(inSCE) elements, or a character scalar that specifies a column name of colData(inSCE). Default NULL.
classGroup1	a vector specifying which "levels" given in class are of interests. Default NULL.
classGroup2	a vector specifying which "levels" given in class is the control group against those specified by classGroup1. If NULL when using annotation specification, classGroup1 cells will be compared with all other cells.
analysisName	A character scalar naming the DEG analysis. Required
groupName1	A character scalar naming the group of interests. Required.
groupName2	A character scalar naming the control group. Required.
covariates	A character vector of additional covariates to use when building the model. All covariates must exist in names(colData(inSCE)). Default NULL.
onlyPos	Whether to only output DEG with positive log2_FC value. Default FALSE.
log2fcThreshold	Only out put DEGs with the absolute values of log2FC greater than this value. Default 0.25
fdrThreshold	Only out put DEGs with FDR value less than this value. Default 0.05
overwrite	A logical scalar. Whether to overwrite result if exists. Default FALSE.

Value

The input SingleCellExperiment object with metadata(inSCE)$diffExp updated with the results: a list named by analysisName, with $groupNames containing the naming of the two conditions, $useAssay storing the assay name that was used for calculation, $select storing the cell selection indices (logical) for each condition, $result storing a data.frame of the DEGs summary, and $method storing "Limma".

Examples

```
data(scExample, package = "singleCellTK")
sce <- subsetSCECols(sce, colData = "type != 'EmptyDroplet'")
sce <- scaterlogNormCounts(sce, assayName = "logcounts")
sce <- runLimmaDE(inSCE = sce, groupName1 = "Sample1",
 groupName2 = "Sample2", index1 = seq(20), index2 = seq(21,40),
 analysisName = "Limma")
```

runMAST *Perform differential expression analysis on SCE with MAST*

Description

Condition specification allows two methods: 1. Index level selection. Arguments `index1` and `index2` will be used. 2. Annotation level selection. Arguments `class`, `classGroup1` and `classGroup2` will be used.

Usage

```
runMAST(
  inSCE,
  useAssay = "logcounts",
  index1 = NULL,
  index2 = NULL,
  class = NULL,
  classGroup1 = NULL,
  classGroup2 = NULL,
  analysisName,
  groupName1,
  groupName2,
  covariates = NULL,
  onlyPos = FALSE,
  log2fcThreshold = NULL,
  fdrThreshold = 0.05,
  overwrite = FALSE,
  check_sanity = TRUE
)
```

Arguments

`inSCE`	SingleCellExperiment inherited object.
`useAssay`	character. A string specifying which assay to use for MAST. The assay should be a log-transformed normalized assay. Default `"logcounts"`.
`index1`	Any type of indices that can subset a SingleCellExperiment inherited object by cells. Specifies which cells are of interests. Default `NULL`.
`index2`	Any type of indices that can subset a SingleCellExperiment inherited object by cells. specifies the control group against those specified by `index1`. If `NULL` when using index specification, `index1` cells will be compared with all other cells. Default `NULL`.
`class`	A vector/factor with `ncol(inSCE)` elements, or a character scalar that specifies a column name of `colData(inSCE)`. Default `NULL`.
`classGroup1`	a vector specifying which "levels" given in `class` are of interests. Default `NULL`.

classGroup2	a vector specifying which "levels" given in class is the control group against those specified by classGroup1. If NULL when using annotation specification, classGroup1 cells will be compared with all other cells.
analysisName	A character scalar naming the DEG analysis. Required
groupName1	A character scalar naming the group of interests. Required.
groupName2	A character scalar naming the control group. Required.
covariates	A character vector of additional covariates to use when building the model. All covariates must exist in names(colData(inSCE)). Default NULL.
onlyPos	Whether to only output DEG with positive log2_FC value. Default FALSE.
log2fcThreshold	Only out put DEGs with the absolute values of log2FC greater than this value. Default 0.25
fdrThreshold	Only out put DEGs with FDR value less than this value. Default 0.05
overwrite	A logical scalar. Whether to overwrite result if exists. Default FALSE.
check_sanity	Logical scalar. Whether to perform MAST's sanity check to see if the counts are logged. Default TRUE.

Value

The input SingleCellExperiment object with metadata(inSCE)$diffExp updated with the results: a list named by analysisName, with $groupNames containing the naming of the two conditions, $useAssay storing the assay name that was used for calculation, $select storing the cell selection indices (logical) for each condition, $result storing a data.frame of the DEGs summary, and $method storing "MAST".

Examples

```
data(scExample, package = "singleCellTK")
sce <- subsetSCECols(sce, colData = "type != 'EmptyDroplet'")
sce <- scaterlogNormCounts(sce[,seq(20)], assayName = "logcounts")
sce <- runMAST(inSCE = sce, groupName1 = "Sample1",
 groupName2 = "Sample2", index1 = seq(10), index2 = seq(11,20),
 analysisName = "MAST")
```

---

runMNNCorrect *Apply the mutual nearest neighbors (MNN) batch effect correction method to SingleCellExperiment object*

Description

MNN is designed for batch correction of single-cell RNA-seq data where the batches are partially confounded with biological conditions of interest. It does so by identifying pairs of MNN in the high-dimensional log-expression space. For each MNN pair, a pairwise correction vector is computed by applying a Gaussian smoothing kernel with bandwidth 'sigma'.

Usage

```
runMNNCorrect(
  inSCE,
  useAssay = "logcounts",
  batch = "batch",
  assayName = "MNN",
  k = 20L,
  sigma = 0.1
)
```

Arguments

inSCE	SingleCellExperiment inherited object. Required.
useAssay	A single character indicating the name of the assay requiring batch correction. Default "logcounts".
batch	A single character indicating a field in colData that annotates the batches. Default "batch".
assayName	A single characeter. The name for the corrected assay. Will be saved to assay. Default "MNN".
k	An integer. Specifies the number of nearest neighbours to consider when defining MNN pairs. This should be interpreted as the minimum frequency of each cell type or state in each batch. Larger values will improve the precision of the correction by increasing the number of MNN pairs, at the cost of reducing accuracy by allowing MNN pairs to form between cells of different type. Default 20L.
sigma	A Numeric scalar. Specifies how much information is shared between MNN pairs when computing the batch effect. Larger values will share more information, approaching a global correction for all cells in the same batch. Smaller values allow the correction to vary across cell types, which may be more accurate but comes at the cost of precision. Default 0.1.

Value

The input SingleCellExperiment object with assay(inSCE,assayName) updated.

References

Lun ATL, et al., 2016 & 2018

Examples

```
data('sceBatches', package = 'singleCellTK')
logcounts(sceBatches) <- log(counts(sceBatches) + 1)
sceCorr <- runMNNCorrect(sceBatches)
```

runNormalization *Wrapper function to run any of the integrated normalization/transformation methods in the singleCellTK. The available methods include 'LogNormalize', 'CLR', 'RC' and 'SCTransform' from Seurat, 'logNormCounts and 'CPM' from Scater. Additionally, users can 'scale' using Z.Score, 'transform' using log, log1p and sqrt, add 'pseudocounts' and trim the final matrices between a range of values.*

Description

Wrapper function to run any of the integrated normalization/transformation methods in the singleCellTK. The available methods include 'LogNormalize', 'CLR', 'RC' and 'SCTransform' from Seurat, 'logNormCounts and 'CPM' from Scater. Additionally, users can 'scale' using Z.Score, 'transform' using log, log1p and sqrt, add 'pseudocounts' and trim the final matrices between a range of values.

Usage

```
runNormalization(
  inSCE,
  useAssay = "counts",
  outAssayName = "customNormalizedAssay",
  normalizationMethod = NULL,
  scale = FALSE,
  seuratScaleFactor = 10000,
  transformation = NULL,
  pseudocountsBeforeNorm = NULL,
  pseudocountsBeforeTransform = NULL,
  trim = NULL,
  verbose = TRUE
)
```

Arguments

inSCE	Input SingleCellExperiment object.
useAssay	Specify the name of the assay that should be used.
outAssayName	Specify the name of the new output assay.
normalizationMethod	Specify a normalization method from 'LogNormalize', 'CLR', 'RC' and 'SCTransform' from Seurat or 'logNormCounts' and 'CPM' from scater packages. Default NULL is set which will not run any normalization method.
scale	Logical value indicating if the data should be scaled using Z.Score. Default FALSE.
seuratScaleFactor	Specify the 'scaleFactor' argument if a Seurat normalization method is selected. Default is 10000. This parameter will not be used if methods other than seurat are selected.

`transformation` Specify the transformation options to run on the selected assay. Options include 'log2' (base 2 log transformation), 'log1p' (natural log + 1 transformation) and 'sqrt' (square root). Default value is `NULL`, which will not run any transformation.

`pseudocountsBeforeNorm` Specify a numeric pseudo value that should be added to the assay before normalization is performed. Default is `NULL`, which will not add any value.

`pseudocountsBeforeTransform` Specify a numeric pseudo value that should be added to the assay before transformation is run. Default is `NULL`, which will not add any value.

`trim` Specify a vector of two numeric values that should be used as the upper and lower trim values to trim the assay between these two values. For example, `c(10,-10)` will trim the values between 10 and -10. Default is `NULL`, which will not trim the data assay.

`verbose` Logical value indicating if progress messages should be displayed to the user. Default is `TRUE`.

Value

Output SCE object with new normalized/transformed assay stored.

Examples

```
data(sce_chcl, package = "scds")
sce_chcl <- runNormalization(
 inSCE = sce_chcl,
 normalizationMethod = "LogNormalize",
 useAssay = "counts",
 outAssayName = "logcounts")
```

`runPerCellQC` *Wrapper for calculating QC metrics with scater.*

Description

A wrapper function for addPerCellQC. Calculate general quality control metrics for each cell in the count matrix.

Usage

```
runPerCellQC(
  inSCE,
  useAssay = "counts",
  collectionName = NULL,
  geneSetList = NULL,
  geneSetListLocation = "rownames",
  geneSetCollection = NULL,
```

```
  percent_top = c(50, 100, 200, 500),
  use_altexps = FALSE,
  flatten = TRUE,
  detectionLimit = 0,
  BPPARAM = BiocParallel::SerialParam()
)
```

Arguments

inSCE	Input SingleCellExperiment object.
useAssay	A string specifying which assay in the SCE to use. Default "counts".
collectionName	Character. Name of a GeneSetCollection obtained by using one of the importGeneSet* functions. Default NULL.
geneSetList	List of gene sets to be quantified. The genes in the assays will be matched to the genes in the list based on geneSetListLocation. Default NULL.
geneSetListLocation	Character or numeric vector. If set to 'rownames', then the genes in 'geneSetList' will be looked up in rownames(inSCE). If another character is supplied, then genes will be looked up in the column names of rowData(inSCE). A character vector with the same length as geneSetList can be supplied if the IDs for different gene sets are found in different places, including a mixture of 'rownames' and rowData(inSCE). An integer or integer vector can be supplied to denote the column index in rowData(inSCE). Default 'rownames'.
geneSetCollection	Class of GeneSetCollection from package GSEAbase. The location of the gene IDs in inSCE should be in the description slot of each gene set and should follow the same notation as geneSetListLocation. The function getGmt can be used to read in gene sets from a GMT file. If reading a GMT file, the second column for each gene set should be the description denoting the location of the gene IDs in inSCE. These gene sets will be included with those from geneSetList if both parameters are provided.
percent_top	An integer vector. Each element is treated as a number of top genes to compute the percentage of library size occupied by the most highly expressed genes in each cell.
use_altexps	Logical scalar indicating whether QC statistics should be computed for alternative Experiments in x. If TRUE, statistics are computed for all alternative experiments. Alternatively, an integer or character vector specifying the alternative Experiments to use to compute QC statistics. Alternatively NULL, in which case alternative experiments are not used.
flatten	Logical scalar indicating whether the nested DataFrame-class in the output should be flattened.
detectionLimit	A numeric scalar specifying the lower detection limit for expression.
BPPARAM	A BiocParallelParam object specifying whether the QC calculations should be parallelized.

Value

A SingleCellExperiment object with cell QC metrics added to the colData slot. If `geneSetList` or `geneSetCollection` are provided, then the rownames for each gene set will be saved in `metadata(inSCE)$scater$addPerC`

Examples

```
data(scExample, package = "singleCellTK")
mito.ix = grep("^MT-", rowData(sce)$feature_name)
geneSet <- list("Mito"=rownames(sce)[mito.ix])
sce <- runPerCellQC(sce, geneSetList = geneSet)
```

`runSCANORAMA` *Apply the mutual nearest neighbors (MNN) batch effect correction method to SingleCellExperiment object*

Description

SCANORAMA is analogous to computer vision algorithms for panorama stitching that identify images with overlapping content and merge these into a larger panorama.

Usage

```
runSCANORAMA(
  inSCE,
  useAssay = "logcounts",
  batch = "batch",
  SIGMA = 15,
  ALPHA = 0.1,
  KNN = 20L,
  assayName = "SCANORAMA"
)
```

Arguments

`inSCE`	SingleCellExperiment inherited object. Required.
`useAssay`	A single character indicating the name of the assay requiring batch correction. Scanorama requires a transformed normalized expression assay. Default `"logcounts"`.
`batch`	A single character indicating a field in `colData` that annotates the batches. Default `"batch"`.
`SIGMA`	A numeric scalar. Algorithmic parameter, correction smoothing parameter on Gaussian kernel. Default `15`.
`ALPHA`	A numeric scalar. Algorithmic parameter, alignment score minimum cutoff. Default `0.1`.
`KNN`	An integer. Algorithmic parameter, number of nearest neighbors to use for matching. Default `20L`.
`assayName`	A single characeter. The name for the corrected assay. Will be saved to `assay`. Default `"SCANORAMA"`.

Value

The input SingleCellExperiment object with `assay(inSCE,assayName)` updated.

References

Brian Hie et al, 2019

Examples

```
## Not run:
data('sceBatches', package = 'singleCellTK')
sceBatches <- scaterlogNormCounts(sceBatches)
sceCorr <- runSCANORAMA(sceBatches, "ScaterLogNormCounts")

## End(Not run)
```

`runScDblFinder` *Detect doublet cells using scDblFinder.*

Description

A wrapper function for scDblFinder. Identify potential doublet cells based on simulations of putative doublet expression profiles. Generate a doublet score for each cell.

Usage

```
runScDblFinder(
  inSCE,
  sample = NULL,
  useAssay = "counts",
  nNeighbors = 50,
  simDoublets = max(10000, ncol(inSCE)),
  seed = 12345,
  BPPARAM = BiocParallel::SerialParam()
)
```

Arguments

`inSCE`	A SingleCellExperiment object.
`sample`	Character vector. Indicates which sample each cell belongs to. scDblFinder will be run on cells from each sample separately.
`useAssay`	A string specifying which assay in the SCE to use.
`nNeighbors`	Number of nearest neighbors used to calculate density for doublet detection. Default 50.
`simDoublets`	Number of simulated doublets created for doublet detection. Default 10000.
`seed`	Seed for the random number generator. Default 12345.
`BPPARAM`	A `BiocParallelParam` object specifying whether the neighbour searches should be parallelized.

Details

This function is a wrapper function for scDblFinder. `runScDblFinder` runs scDblFinder for each `sample` within `inSCE` iteratively. The resulting doublet scores for all cells will be appended to the colData of `inSCE`.

Value

A SingleCellExperiment object with the scDblFinder QC outputs added to the colData slot.

References

Lun ATL (2018). Detecting doublet cells with scran. https://ltla.github.io/SingleCellThoughts/software/doublet_detection/bycell.html

See Also

scDblFinder

Examples

```
data(scExample, package = "singleCellTK")
sce <- subsetSCECols(sce, colData = "type != 'EmptyDroplet'")
sce <- runScDblFinder(sce)
```

`runSCMerge` *Apply scMerge batch effect correction method to SingleCellExperiment object*

Description

The scMerge method leverages factor analysis, stably expressed genes (SEGs) and (pseudo-) replicates to remove unwanted variations and merge multiple scRNA-Seq data.

Usage

```
runSCMerge(
  inSCE,
  useAssay = "logcounts",
  batch = "batch",
  assayName = "scMerge",
  seg = NULL,
  kmeansK = NULL,
  cellType = "cell_type",
  nCores = 1L
)
```

Arguments

inSCE	SingleCellExperiment inherited object. Required.
useAssay	A single character indicating the name of the assay requiring batch correction. Default "logcounts".
batch	A single character indicating a field in colData that annotates the batches. Default "batch".
assayName	A single characeter. The name for the corrected assay. Will be saved to assay. Default "scMerge".
seg	A vector of gene names or indices that specifies SEG (Stably Expressed Genes) set as negative control. Pre-defined dataset with human and mouse SEG lists is available to user by running data('SEG'). Default NULL, and this value will be auto-detected by default with scSEGIndex.
kmeansK	An integer vector. Indicating the kmeans' K-value for each batch (i.e. how many subclusters in each batch should exist), in order to construct pseudo-replicates. The length of codekmeansK needs to be the same as the number of batches. Default NULL, and this value will be auto-detected by default, depending on cellType.
cellType	A single character. A string indicating a field in colData(inSCE) that defines different cell types. Default 'cell_type'.
nCores	An integer. The number of cores of processors to allocate for the task. Default 1L.

Value

The input SingleCellExperiment object with assay(inSCE,assayName) updated.

References

Hoa, et al., 2020

Examples

```
data('sceBatches', package = 'singleCellTK')
## Not run:
logcounts(sceBatches) <- log(counts(sceBatches) + 1)
sceCorr <- runSCMerge(sceBatches)

## End(Not run)
```

runScranSNN *Get clustering with SNN graph*

Description

Perform SNN graph clustering on a SingleCellExperiment object, with graph construction by buildSNNGraph and graph clustering by "igraph" package.

Usage

```
runScranSNN(
  inSCE,
  useAssay = NULL,
  useReducedDim = NULL,
  useAltExp = NULL,
  altExpAssay = "counts",
  altExpRedDim = NULL,
  clusterName = "scranSNN_cluster",
  k = 10,
  nComp = 50,
  weightType = c("rank", "number", "jaccard"),
  algorithm = c("walktrap", "louvain", "infomap", "fastGreedy", "labelProp",
    "leadingEigen")
)
```

Arguments

inSCE	A SingleCellExperiment object.
useAssay	A single character, specifying which assay to perform the clustering algorithm on. Default NULL.
useReducedDim	A single character, specifying which low-dimension representation (reducedDim) to perform the clustering algorithm on. Default NULL.
useAltExp	A single character, specifying the assay which altExp to perform the clustering algorithm on. Default NULL.
altExpAssay	A single character, specifying which assay in the chosen altExp to work on. Only used when useAltExp is set. Default "counts".
altExpRedDim	A single character, specifying which reducedDim within the altExp specified by useAltExp to use. Only used when useAltExp is set. Default NULL.
clusterName	A single character, specifying the name to store the cluster label in colData. Default "scranSNN_cluster".
k	An integer, the number of nearest neighbors used to construct the graph. Smaller value indicates higher resolution and larger number of clusters. Default 10.
nComp	An integer, the number of components to use when useAssay or useAltExp is specified. WON'T work with useReducedDim. Default 50.

weightType	A single character, that specifies the edge weighing scheme when constructing the Shared Nearest-Neighbor (SNN) graph. Choose from "rank", "number", "jaccard". Default "rank".
algorithm	A single character, that specifies the community detection algorithm to work on the SNN graph. Choose from "walktrap", "louvain", "infomap", "fastGreedy", "labelProp", "leadingEigen". Default "walktrap".

Value

The input SingleCellExperiment object with factor cluster labeling updated in colData(inSCE)[[clusterName]].

References

Aaron Lun and et. al., 2016

Examples

```
data("mouseBrainSubsetSCE")
mouseBrainSubsetSCE <- runScranSNN(mouseBrainSubsetSCE,
                                   useReducedDim = "PCA_logcounts")
```

---

runScrublet *Find doublets using* scrublet.

Description

A wrapper function that calls scrub_doublets from python module scrublet. Simulates doublets from the observed data and uses a k-nearest-neighbor classifier to calculate a continuous scrublet_score (between 0 and 1) for each transcriptome. The score is automatically thresholded to generate scrublet_call, a boolean array that is TRUE for predicted doublets and FALSE otherwise.

Usage

```
runScrublet(
  inSCE,
  sample = NULL,
  useAssay = "counts",
  simDoubletRatio = 2,
  nNeighbors = NULL,
  minDist = NULL,
  expectedDoubletRate = 0.1,
  stdevDoubletRate = 0.02,
  syntheticDoubletUmiSubsampling = 1,
  useApproxNeighbors = TRUE,
  distanceMetric = "euclidean",
  getDoubletNeighborParents = FALSE,
```

```
  minCounts = 3,
  minCells = 3L,
  minGeneVariabilityPctl = 85,
  logTransform = FALSE,
  meanCenter = TRUE,
  normalizeVariance = TRUE,
  nPrinComps = 30L,
  tsneAngle = NULL,
  tsnePerplexity = NULL,
  verbose = TRUE,
  seed = 12345
)
```

Arguments

inSCE	A SingleCellExperiment object. Needs counts in assays slot.
sample	Character vector. Indicates which sample each cell belongs to. Scrublet will be run on cells from each sample separately. If NULL, then all cells will be processed together. Default NULL.
useAssay	A string specifying which assay in the SCE to use. Default 'counts'.
simDoubletRatio	Numeric. Number of doublets to simulate relative to the number of observed transcriptomes. Default 2.0.
nNeighbors	Integer. Number of neighbors used to construct the KNN graph of observed transcriptomes and simulated doublets. If NULL, this is set to round(0.5 * sqrt(n_cells)). Default NULL.
minDist	Float Determines how tightly UMAP packs points together. If NULL, this is set to 0.1. Default NULL.
expectedDoubletRate	The estimated doublet rate for the experiment. Default 0.1.
stdevDoubletRate	Uncertainty in the expected doublet rate. Default 0.02.
syntheticDoubletUmiSubsampling	Numeric. Rate for sampling UMIs when creating synthetic doublets. If 1.0, each doublet is created by simply adding the UMIs from two randomly sampled observed transcriptomes. For values less than 1, the UMI counts are added and then randomly sampled at the specified rate. Defuault: 1.0.
useApproxNeighbors	Boolean. Use approximate nearest neighbor method (annoy) for the KNN classifier. Default TRUE.
distanceMetric	Character. Distance metric used when finding nearest neighbors. For list of valid values, see the documentation for annoy (if useApproxNeighbors is TRUE) or sklearn.neighbors.NearestNeighbors (if useApproxNeighbors is FALSE). Default "euclidean".
getDoubletNeighborParents	Boolean. If TRUE, return the parent transcriptomes that generated the doublet neighbors of each observed transcriptome. This information can be used to infer the cell states that generated a given doublet state. Default FALSE.

minCounts — Numeric. Used for gene filtering prior to PCA. Genes expressed at fewer than minCounts in fewer than minCells (see below) are excluded. Default 3.

minCells — Integer. Used for gene filtering prior to PCA. Genes expressed at fewer than minCounts (see above) in fewer than minCells are excluded. Default 3.

minGeneVariabilityPctl — Numeric. Used for gene filtering prior to PCA. Keep the most highly variable genes (in the top minGeneVariabilityPctl percentile), as measured by the v-statistic (*Klein et al., Cell 2015*). Default 85.

logTransform — Boolean. If TRUE, log-transform the counts matrix (log10(1+TPM)). sklearn.decomposition.Truncate will be used for dimensionality reduction, unless meanCenter is TRUE. Default FALSE.

meanCenter — If TRUE, center the data such that each gene has a mean of 0. sklearn.decomposition.PCA will be used for dimensionality reduction. Default TRUE.

normalizeVariance — Boolean. If TRUE, normalize the data such that each gene has a variance of 1. sklearn.decomposition.TruncatedSVD will be used for dimensionality reduction, unless meanCenter is TRUE. Default TRUE.

nPrinComps — Integer. Number of principal components used to embed the transcriptomes prior to k-nearest-neighbor graph construction. Default 30.

tsneAngle — Float. Determines angular size of a distant node as measured from a point in the t-SNE plot. If default, it is set to 0.5 Default NULL.

tsnePerplexity — Integer. The number of nearest neighbors that is used in other manifold learning algorithms. If default, it is set to 30. Default NULL.

verbose — Boolean. If TRUE, print progress updates. Default TRUE.

seed — Seed for the random number generator. Default 12345.

Value

A SingleCellExperiment object with scrub_doublets output appended to the colData slot. The columns include *scrublet_score* and *scrublet_call*.

Examples

```
data(scExample, package = "singleCellTK")
## Not run:
sce <- subsetSCECols(sce, colData = "type != 'EmptyDroplet'")
sce <- runScrublet(sce)

## End(Not run)
```

runSingleR *Label cell types with SingleR*

Description

SingleR works with a reference dataset where the cell type labeling is given. Given a reference dataset of samples (single-cell or bulk) with known labels, it assigns those labels to new cells from a test dataset based on similarities in their expression profiles.

Usage

```
runSingleR(
  inSCE,
  useAssay = "logcounts",
  useSCERef = NULL,
  labelColName = NULL,
  useBltinRef = c("hpca", "bpe", "mp", "dice", "immgen", "mouse", "zeisel"),
  level = c("main", "fine", "ont"),
  featureType = c("symbol", "ensembl"),
  labelByCluster = NULL
)
```

Arguments

inSCE	SingleCellExperiment inherited object. Required.
useAssay	character. A string specifying which assay to use for expression profile identification. Required.
useSCERef	SingleCellExperiment inherited object. An optional customized reference dataset. Default NULL.
labelColName	A single character. A string specifying the column in colData(useSCERef) that stores the cell type labeling. Default NULL.
useBltinRef	A single character. A string that specifies a reference provided by SingleR. Choose from "hpca","bpe","mp","dice","immgen","mouse","zeisel". See detail. Default "hpca".
level	A string for cell type labeling level. Used only when using some of the SingleR built-in references. Choose from "main","fine","ont". Default "main".
featureType	A string for whether to use gene symbols or Ensembl IDs when using a SingleR built-in reference. Should be set based on the type of rownames of inSCE. Choose from "symbol","ensembl". Default "symbol".
labelByCluster	A single character. A string specifying the column name in colData(inSCE) that stores clustering labels. Use this when users want to only label cells on cluster level, instead of performing calculation on each cell. Default NULL.

Value

Input SCE object with cell type labeling updated in colData(inSCE), together with scoring metrics.

Examples

```
data("sceBatches")
logcounts(sceBatches) <- log(counts(sceBatches) + 1)
#sceBatches <- runSingleR(sceBatches, useBltinRef = "mp")
```

---

runVAM *Run VAM to score gene sets in single cell data*

Description

Wrapper for the Variance-adjusted Mahalanobis (VAM), which is a fast and accurate method for cell-specific gene set scoring of single cell data. This algorithm computes distance statistics and one-sided p-values for all cells in the specified single cell gene expression matrix. Gene sets should already be imported and stored in the meta data using functions such as importGeneSetsFromList or importGeneSetsFromMSigDB

Usage

```
runVAM(
  inSCE,
  geneSetCollectionName,
  useAssay,
  resultNamePrefix = NULL,
  center = TRUE,
  gamma = FALSE
)
```

Arguments

inSCE	Input SingleCellExperiment object.
geneSetCollectionName	Character. The name of the gene set collection to use.
useAssay	Character. The name of the assay to use. This assay should contain log normalized counts.
resultNamePrefix	Character. Prefix to the name the VAM results which will be stored in the reducedDim slot of inSCE. The names of the output matrices will be resultNamePrefix_Distance and resultNamePrefix_CDF. If this parameter is set to NULL, then "VAM_geneSetCollectionName_" will be used. Default NULL.
center	Boolean. If TRUE, values will be mean centered when computating the Mahalanobis statistic. Default TRUE.
gamma	Boolean. If TRUE, a gamma distribution will be fit to the non-zero squared Mahalanobis distances computed from a row-permuted version of the gene expression matrix. The estimated gamma distribution will be used to compute a one-sided p-value for each cell. If FALSE, the p-value will be computed using the standard chi-square approximation for the squared Mahalanobis distance (or non-central if center = FALSE). Default FALSE.

Value

A SingleCellExperiment object with VAM metrics stored in `reducedDim` as `VAM_NameOfTheGeneset_Distance` and `VAM_NameOfTheGeneset_CDF`.

Author(s)

Nida Pervaiz

See Also

importGeneSetsFromList, importGeneSetsFromMSigDB, importGeneSetsFromGMT, importGeneSetsFromCollection for importing gene sets.

Examples

```
data(scExample, package = "singleCellTK")
sce <- subsetSCECols(sce, colData = "type != 'EmptyDroplet'")
sce <- scaterlogNormCounts(sce, assayName = "logcounts")
gs1 <- rownames(sce)[seq(10)]
gs2 <- rownames(sce)[seq(11,20)]
gs <- list("geneset1" = gs1, "geneset2" = gs2)

sce <- importGeneSetsFromList(inSCE = sce,geneSetList = gs,
                                            by = "rownames")
sce <- runVAM(inSCE = sce, geneSetCollectionName = "GeneSetCollection", useAssay = "logcounts")
```

---

runWilcox *Perform differential expression analysis on SCE with Wilcoxon test*

Description

Condition specification allows two methods: 1. Index level selection. Arguments `index1` and `index2` will be used. 2. Annotation level selection. Arguments `class`, `classGroup1` and `classGroup2` will be used.

Usage

```
runWilcox(
  inSCE,
  useAssay = "logcounts",
  index1 = NULL,
  index2 = NULL,
  class = NULL,
  classGroup1 = NULL,
  classGroup2 = NULL,
  analysisName,
```

```
  groupName1,
  groupName2,
  covariates = NULL,
  onlyPos = FALSE,
  log2fcThreshold = 0.25,
  fdrThreshold = 0.05,
  overwrite = FALSE
)
```

Arguments

inSCE	SingleCellExperiment inherited object.
useAssay	character. A string specifying which assay to use for the Wilcoxon test. The assay should be a log-transformed normalized assay. Default "logcounts".
index1	Any type of indices that can subset a SingleCellExperiment inherited object by cells. Specifies which cells are of interests. Default NULL.
index2	Any type of indices that can subset a SingleCellExperiment inherited object by cells. specifies the control group against those specified by index1. If NULL when using index specification, index1 cells will be compared with all other cells. Default NULL.
class	A vector/factor with ncol(inSCE) elements, or a character scalar that specifies a column name of colData(inSCE). Default NULL.
classGroup1	a vector specifying which "levels" given in class are of interests. Default NULL.
classGroup2	a vector specifying which "levels" given in class is the control group against those specified by classGroup1. If NULL when using annotation specification, classGroup1 cells will be compared with all other cells.
analysisName	A character scalar naming the DEG analysis. Required
groupName1	A character scalar naming the group of interests. Required.
groupName2	A character scalar naming the control group. Required.
covariates	Not supported by pairwiseWilcox, will be ignored if any, but included in metadata for plotting. Default NULL.
onlyPos	Whether to only output DEG with positive log2_FC value. Default FALSE.
log2fcThreshold	Only out put DEGs with the absolute values of log2FC greater than this value. Default 0.25
fdrThreshold	Only out put DEGs with FDR value less than this value. Default 0.05
overwrite	A logical scalar. Whether to overwrite result if exists. Default FALSE.

Value

The input SingleCellExperiment object with metadata(inSCE)$diffExp updated with the results: a list named by analysisName, with $groupNames containing the naming of the two conditions, $useAssay storing the assay name that was used for calculation, $select storing the cell selection indices (logical) for each condition, $result storing a data.frame of the DEGs summary, and $method storing "wilcox".

Examples

```
data(scExample, package = "singleCellTK")
sce <- subsetSCECols(sce, colData = "type != 'EmptyDroplet'")
sce <- scaterlogNormCounts(sce, assayName = "logcounts")
sce <- runWilcox(inSCE = sce, groupName1 = "Sample1",
 groupName2 = "Sample2", index1 = seq(20), index2 = seq(21,40),
 analysisName = "wilcox")
```

`runZINBWaVE` *Apply ZINBWaVE Batch effect correction method to SingleCellExperiment object*

Description

A general and flexible zero-inflated negative binomial model that can be used to provide a low-dimensional representations of scRNAseq data. The model accounts for zero inflation (dropouts), over-dispersion, and the count nature of the data. The model also accounts for the difference in library sizes and optionally for batch effects and/or other covariates.

Usage

```
runZINBWaVE(
  inSCE,
  useAssay = "counts",
  batch = "batch",
  nHVG = 1000L,
  nComponents = 50L,
  epsilon = 1000,
  nIter = 10L,
  reducedDimName = "zinbwave"
)
```

Arguments

`inSCE`	SingleCellExperiment inherited object. Required.
`useAssay`	A single character indicating the name of the assay requiring batch correction. Note that ZINBWaVE works for counts (integer) input rather than logcounts that other methods prefer. Default `"counts"`.
`batch`	A single character indicating a field in `colData` that annotates the batches. Default `"batch"`.
`nHVG`	An integer. Number of highly variable genes to use when fitting the model. Default `1000L`.
`nComponents`	An integer. The number of principle components or dimensionality to generate in the resulting matrix. Default `50L`.
`epsilon`	An integer. Algorithmic parameter. Empirically, a high epsilon is often required to obtained a good low-level representation. Default `1000L`.

nIter	An integer, The max number of iterations to perform. Default 10L.
reducedDimName	A single character. The name for the corrected low-dimensional representation. Will be saved to reducedDim(inSCE). Default "zinbwave".

Value

The input SingleCellExperiment object with reducedDim(inSCE,reducedDimName) updated.

References

Pollen, Alex A et al., 2014

Examples

```
data('sceBatches', package = 'singleCellTK')
## Not run:
    sceCorr <- runZINBWaVE(sceBatches, nIter = 5)

## End(Not run)
```

---

sampleSummaryStats *Generate table of SCTK QC outputs.*

Description

Creates a table of QC metrics generated from QC algorithms via either kable or csv file.

Usage

```
sampleSummaryStats(inSCE, sample = NULL, useAssay = "counts", simple = TRUE)
```

Arguments

inSCE	Input SingleCellExperiment object with saved assay data and/or colData data. Required.
sample	Character vector. Indicates which sample each cell belongs to.
useAssay	A string specifying which assay in the SCE to use. Default 'counts'.
simple	Boolean. Indicates whether to generate a table of only basic QC stats (ex. library size), or to generate a summary table of all QC stats stored in the inSCE.

Value

A matrix/array object.

Examples

```
data(scExample, package = "singleCellTK")
sce <- subsetSCECols(sce, colData = "type != 'EmptyDroplet'")
sampleSummaryStats(sce, simple = TRUE)
```

`scaterCPM` *scaterCPM Uses CPM from scater library to compute counts-per-million.*

Description

scaterCPM Uses CPM from scater library to compute counts-per-million.

Usage

```
scaterCPM(inSCE, assayName = "ScaterCPMCounts", useAssay = "counts")
```

Arguments

`inSCE`	Input SingleCellExperiment object
`assayName`	New assay name for cpm data.
`useAssay`	Input assay

Value

inSCE Updated SingleCellExperiment object

Author(s)

Irzam Sarfraz

Examples

```
data(sce_chcl, package = "scds")
sce_chcl <- scaterCPM(sce_chcl,"countsCPM", "counts")
```

`scaterlogNormCounts` *scaterlogNormCounts Uses logNormCounts to log normalize input data*

Description

scaterlogNormCounts Uses logNormCounts to log normalize input data

Usage

```
scaterlogNormCounts(
  inSCE,
  assayName = "ScaterLogNormCounts",
  useAssay = "counts"
)
```

Arguments

inSCE	Input SingleCellExperiment object
assayName	New assay name for log normalized data
useAssay	Input assay

Value

inSCE Updated SingleCellExperiment object that contains the new log normalized data

Author(s)

Irzam Sarfraz

Examples

```
data(sce_chcl, package = "scds")
sce_chcl <- scaterlogNormCounts(sce_chcl,"logcounts", "counts")
```

---

scaterPCA *Perform PCA on a SingleCellExperiment Object A wrapper to runPCA function to compute principal component analysis (PCA) from a given SingleCellExperiment object.*

Description

Perform PCA on a SingleCellExperiment Object A wrapper to runPCA function to compute principal component analysis (PCA) from a given SingleCellExperiment object.

Usage

```
scaterPCA(
  inSCE,
  useAssay = "logcounts",
  useAltExp = NULL,
  reducedDimName = "PCA",
  nComponents = 50,
  scale = FALSE,
  ntop = NULL
)
```

Arguments

inSCE	Input SingleCellExperiment object.
useAssay	Assay to use for PCA computation. If useAltExp is specified, useAssay has to exist in assays(altExp(inSCE,useAltExp)). Default "logcounts"

useAltExp	The subset to use for PCA computation, usually for the selected.variable features. Default NULL.
reducedDimName	Name to use for the reduced output assay. Default "PCA".
nComponents	Number of principal components to obtain from the PCA computation. Default 50.
scale	Logical scalar, whether to standardize the expression values. Default FALSE.
ntop	Number of top features to use as a further variable feature selection. Default NULL.

Value

A SingleCellExperiment object with PCA computation updated in reducedDim(inSCE,reducedDimName).

Examples

```
data(scExample, package = "singleCellTK")
sce <- subsetSCECols(sce, colData = "type != 'EmptyDroplet'")
sce <- scaterlogNormCounts(sce, "logcounts")
sce <- scaterPCA(sce, "logcounts")
```

---

sce *Example Single Cell RNA-Seq data in SingleCellExperiment Object, subset of 10x public dataset https://support.10xgenomics.com/single-cell-gene-expression/datasets/2.1.0/pbmc4k A subset of 390 barcodes and top 200 genes were included in this example. Within 390 barcodes, 195 barcodes are empty droplet, 150 barcodes are cell barcode and 45 barcodes are doublets predicted by scrublet and doubletFinder package. This example only serves as a proof of concept and a tutoriol on how to run the functions in this package. The results should not be used for drawing scientific conclusions.*

Description

Example Single Cell RNA-Seq data in SingleCellExperiment Object, subset of 10x public dataset https://support.10xgenomics.com/single-cell-gene-expression/datasets/2.1.0/pbmc4k A subset of 390 barcodes and top 200 genes were included in this example. Within 390 barcodes, 195 barcodes are empty droplet, 150 barcodes are cell barcode and 45 barcodes are doublets predicted by scrublet and doubletFinder package. This example only serves as a proof of concept and a tutoriol on how to run the functions in this package. The results should not be used for drawing scientific conclusions.

Usage

```
sce
```

Format

A SingleCellExperiment object.

Examples

```
data("scExample")
```

`sceBatches` *Example Single Cell RNA-Seq data in SingleCellExperiment object, with different batches annotated*

Description

Two batches of pancreas scRNAseq dataset are combined with their original counts. Cell types and batches are annotated in 'colData(sceBatches)'. Two batches came from Wang, et al., 2016, annotated as ''w''; and Xin, et al., 2016, annotated as ''x''. Two common cell types, ''alpha'' and ''beta'', that could be found in both original studies with relatively large population were kept for cleaner demonstration. data('sceBatches')

Usage

```
sceBatches
```

Format

An object of class `SingleCellExperiment` with 100 rows and 250 columns.

`scranModelGeneVar` *scranModelGeneVar Generates and stores variability data from scran::modelGeneVar in the input singleCellExperiment object*

Description

scranModelGeneVar Generates and stores variability data from scran::modelGeneVar in the input singleCellExperiment object

Usage

```
scranModelGeneVar(inSCE, assayName)
```

Arguments

`inSCE`	a singleCellExperiment object
`assayName`	selected assay to compute variable features from

Value

inSCE updated singleCellExperiment object that contains variable feature metrics in rowData

Author(s)

Irzam Sarfraz

Examples

```
data(sce_chcl, package = "scds")
sce_chcl <- scranModelGeneVar(sce_chcl, "counts")
```

---

sctkListGeneSetCollections

Lists imported GeneSetCollections

Description

Returns a vector of GeneSetCollections that have been imported and stored in `metadata(inSCE)$sctk$genesets`.

Usage

```
sctkListGeneSetCollections(inSCE)
```

Arguments

`inSCE`	A SingleCellExperiment object.

Value

Character vector.

Author(s)

Joshua D. Campbell

See Also

importGeneSetsFromList for importing from lists, importGeneSetsFromGMT for importing from GMT files, GeneSetCollection objects, and importGeneSetsFromMSigDB for importing MSigDB gene sets.

Examples

```
data(scExample)
library(GSEABase)
gs1 <- GeneSet(setName = "geneset1", geneIds = rownames(sce)[seq(10)])
gs2 <- GeneSet(setName = "geneset2", geneIds = rownames(sce)[seq(11,20)])
gsc1 <- GeneSetCollection(gs1)
gsc2 <- GeneSetCollection(gs2)
sce <- importGeneSetsFromCollection(inSCE = sce,
                                    geneSetCollection = gsc1,
                                    by = "rownames",
```

```
                                    collectionName = "Collection1")
sce <- importGeneSetsFromCollection(inSCE = sce,
                                    geneSetCollection = gsc2,
                                    by = "rownames",
                                    collectionName = "Collection2")
collections <- sctkListGeneSetCollections(sce)
```

`sctkPythonInstallConda` *Installs Python packages into a Conda environment*

Description

Install all Python packages used in the `singleCellTK` package using `conda_install` from package `reticulate`. This will create a new Conda environment with the name `envname` if not already present. Note that Anaconda or Miniconda already need to be installed on the local system.

Usage

```
sctkPythonInstallConda(
  envname = "sctk-reticulate",
  conda = "auto",
  packages = c("scipy", "numpy", "astroid", "six"),
  pipPackages = c("scrublet", "scanpy", "bbknn", "scanorama", "anndata"),
  selectConda = TRUE,
  forge = FALSE,
  pipIgnoreInstalled = TRUE,
  pythonVersion = NULL,
  ...
)
```

Arguments

`envname`	Character. Name of the conda environment to create.
`conda`	Character. Path to conda executable. Usue "auto" to find conda using the PATH and other conventional install locations. Default 'auto'.
`packages`	Character Vector. List of packages to install from Conda.
`pipPackages`	Character Vector. List of packages to install into the Conda environment using 'pip'.
`selectConda`	Boolean. Run `selectSCTKConda` after installing all packages to select the Conda environment. Default TRUE.
`forge`	Boolean. Include the Conda Forge repository.
`pipIgnoreInstalled`	Boolean. Ignore installed versions when using pip. This is TRUE by default so that specific package versions can be installed even if they are downgrades. The FALSE option is useful for situations where you don't want a pip install to

	attempt an overwrite of a conda binary package (e.g. SciPy on Windows which is very difficult to install via pip due to compilation requirements).
pythonVersion	Passed to python_version variable in conda_install. Default NULL.
...	Other parameters to pass to conda_install.

Value

None. Installation of Conda environment.

See Also

See conda_create for more information on creating a Conda environment. See conda_install for more description of the installation parameters. See https://rstudio.github.io/reticulate/ for more information on package reticulate. See selectSCTKConda for reloading the Conda environment if R is restarted without going through the whole installation process again. See https://docs.conda.io/en/latest/ for more information on Conda environments.

Examples

```
## Not run:
sctkPythonInstallConda(envname = "sctk-reticulate")

## End(Not run)
```

---

sctkPythonInstallVirtualEnv

Installs Python packages into a virtual environment

Description

Install all Python packages used in the singleCellTK package using virtualenv_install from package reticulate. This will create a new virtual environment with the name envname if not already present.

Usage

```
sctkPythonInstallVirtualEnv(
  envname = "sctk-reticulate",
 packages = c("scipy", "numpy", "astroid", "six", "scrublet", "scanpy", "scanorama",
    "bbknn", "anndata"),
  selectEnvironment = TRUE,
  python = NULL
)
```

Arguments

envname	Character. Name of the virtual environment to create.
packages	Character Vector. List of packages to install.
selectEnvironment	Boolean. Run selectSCTKVirtualEnvironment after installing all packages to select the virtual environment. Default TRUE.
python	The path to a Python interpreter, to be used with the created virtual environment. When NULL, the Python interpreter associated with the current session will be used. Default NULL.

Value

None. Installation of virtual environment.

See Also

See virtualenv_create for more information on creating a Conda environment. See virtualenv_install for more description of the installation parameters. See https://rstudio.github.io/reticulate/ for more information on package reticulate. See selectSCTKVirtualEnvironment for reloading the virtual environment if R is restarted without going through the whole installation process again.

Examples

```
## Not run:
sctkPythonInstallVirtualEnv(envname = "sctk-reticulate")

## End(Not run)
```

SEG *Stably Expressed Gene (SEG) list obect, with SEG sets for human and mouse.*

Description

The two gene sets came from dataset called ‘segList‘ of package ‘scMerge‘.

Usage

```
SEG
```

Format

list, with two entries "human" and "mouse", each is a charactor vector.

Source

```
data('segList',package='scMerge')
```

Examples

```
data('SEG')
humanSEG <- SEG$human
```

`selectSCTKConda` *Selects a Conda environment*

Description

Selects a Conda environment with Python packages used in singleCellTK.

Usage

```
selectSCTKConda(envname = "sctk-reticulate")
```

Arguments

envname	Character. Name of the conda environment to activate.

Value

None. Selects Conda environment.

See Also

conda-tools for more information on using Conda environments with package reticulate. See https://rstudio.github.io/reticulate/ for more information on package reticulate.

See sctkPythonInstallConda for installation of Python modules into a Conda environment. Seeconda-tools for more information on using Conda environments with package reticulate. See https://rstudio.github.io/reticulate/ for more information on package reticulate. See https://docs.conda.io/en/latest/ for more information on Conda environments.

Examples

```
## Not run:
sctkPythonInstallConda(envname = "sctk-reticulate", selectConda = FALSE)
selectSCTKConda(envname = "sctk-reticulate")

## End(Not run)
```

selectSCTKVirtualEnvironment *Selects a virtual environment*

Description

Selects a virtual environment with Python packages used in singleCellTK

Usage

```
selectSCTKVirtualEnvironment(envname = "sctk-reticulate")
```

Arguments

envname	Character. Name of the virtual environment to activate.

Value

None. Selects virtual environment.

See Also

See sctkPythonInstallVirtualEnv for installation of Python modules into a virtual environment. Seevirtualenv-tools for more information on using virtual environments with package reticulate. See https://rstudio.github.io/reticulate/ for more information on package reticulate.

Examples

```
## Not run:
sctkPythonInstallVirtualEnv(envname = "sctk-reticulate", selectEnvironment = FALSE)
selectSCTKVirtualEnvironment(envname = "sctk-reticulate")

## End(Not run)
```

setSCTKDisplayRow *Indicates which rowData to use for visualization*

Description

This function is to be used to specify which

Usage

```
setSCTKDisplayRow(inSCE, featureDisplayRow)
```

Arguments

inSCE	Input SingleCellExperiment object with saved dimension reduction components or a variable with saved results. Required.
featureDisplayRow	Indicates which column name of rowData to be used for plots.

Value

A SingleCellExperiment object with the specific column name of rowData to be used for plotting stored in metadata.

Examples

```
data(scExample, package="singleCellTK")
sce <- subsetSCECols(sce, colData = "type != 'EmptyDroplet'")
sce <- setSCTKDisplayRow(inSCE = sce, featureDisplayRow = "feature_name")
plotSCEViolinAssayData(inSCE = sce, feature = "ENSG00000019582")
```

---

seuratComputeHeatmap *seuratComputeHeatmap Computes the heatmap plot object from the pca slot in the input sce object*

Description

seuratComputeHeatmap Computes the heatmap plot object from the pca slot in the input sce object

Usage

```
seuratComputeHeatmap(
  inSCE,
  useAssay,
  useReduction = c("pca", "ica"),
  dims = NULL,
  nfeatures = 30,
  cells = NULL,
  ncol = NULL,
  balanced = TRUE,
  fast = TRUE,
  combine = TRUE,
  raster = TRUE,
  externalReduction = NULL
)
```

Arguments

inSCE	(sce) object from which to compute heatmap (pca should be computed)
useAssay	Assay containing scaled counts to use in heatmap.
useReduction	Reduction method to use for computing clusters. One of "pca" or "ica". Default "pca".
dims	Number of components to generate heatmap plot objects. If NULL, a heatmap will be generated for all components. Default NULL.
nfeatures	Number of features to include in the heatmap. Default 30.
cells	Numeric value indicating the number of top cells to plot. Default is NULL which indicates all cells.
ncol	Numeric value indicating the number of columns to use for plot. Default is NULL which will automatically compute accordingly.
balanced	Plot equal number of genes with positive and negative scores. Default is TRUE.
fast	See DimHeatmap for more information. Default TRUE.
combine	See DimHeatmap for more information. Default TRUE.
raster	See DimHeatmap for more information. Default TRUE.
externalReduction	Pass DimReduc object if PCA/ICA computed through other libraries. Default NULL.

Value

plot object

Examples

```
data(scExample, package = "singleCellTK")
## Not run:
sce <- seuratNormalizeData(sce, useAssay = "counts")
sce <- seuratFindHVG(sce, useAssay = "counts")
sce <- seuratScaleData(sce, useAssay = "counts")
sce <- seuratPCA(sce, useAssay = "counts")
heatmap <- seuratComputeHeatmap(sce, useAssay = "counts")
seuratHeatmapPlot(heatmap)

## End(Not run)
```

---

seuratComputeJackStraw

seuratComputeJackStraw Compute jackstraw plot and store the computations in the input sce object

---

Description

seuratComputeJackStraw Compute jackstraw plot and store the computations in the input sce object

Usage

```
seuratComputeJackStraw(
  inSCE,
  useAssay,
  dims = NULL,
  numReplicate = 100,
  propFreq = 0.025,
  externalReduction = NULL
)
```

Arguments

inSCE	(sce) object on which to compute and store jackstraw plot
useAssay	Assay containing scaled counts to use in JackStraw calculation.
dims	Number of components to test in Jackstraw. If NULL, then all components are used. Default NULL.
numReplicate	Numeric value indicating the number of replicate samplings to perform. Default value is 100.
propFreq	Numeric value indicating the proportion of data to randomly permute for each replicate. Default value is 0.025.
externalReduction	Pass DimReduc object if PCA/ICA computed through other libraries. Default NULL.

Value

Updated SingleCellExperiment object with jackstraw computations stored in it

Examples

```
data(scExample, package = "singleCellTK")
## Not run:
sce <- seuratNormalizeData(sce, useAssay = "counts")
sce <- seuratFindHVG(sce, useAssay = "counts")
sce <- seuratScaleData(sce, useAssay = "counts")
sce <- seuratPCA(sce, useAssay = "counts")
sce <- seuratComputeJackStraw(sce, useAssay = "counts")

## End(Not run)
```

---

seuratElbowPlot *seuratElbowPlot Computes the plot object for elbow plot from the pca slot in the input sce object*

Description

seuratElbowPlot Computes the plot object for elbow plot from the pca slot in the input sce object

Usage

```
seuratElbowPlot(
  inSCE,
  significantPC = NULL,
  reduction = "pca",
  ndims = 20,
  externalReduction = NULL,
  interactive = TRUE
)
```

Arguments

inSCE	(sce) object from which to compute the elbow plot (pca should be computed)
significantPC	Number of significant principal components to plot. This is used to alter the color of the points for the corresponding PCs. If NULL, all points will be the same color. Default NULL.
reduction	Reduction to use for elbow plot generation. Either "pca" or "ica". Default "pca".
ndims	Number of components to use. Default 20.
externalReduction	Pass DimReduc object if PCA/ICA computed through other libraries. Default NULL.
interactive	Logical value indicating if the returned object should be an interactive plotly object if TRUE or a ggplot object if set to FALSE. Default is TRUE.

Value

plot object

Examples

```
data(scExample, package = "singleCellTK")
## Not run:
sce <- seuratNormalizeData(sce, useAssay = "counts")
sce <- seuratFindHVG(sce, useAssay = "counts")
sce <- seuratScaleData(sce, useAssay = "counts")
sce <- seuratPCA(sce, useAssay = "counts")
seuratElbowPlot(sce)

## End(Not run)
```

seuratFindClusters *seuratFindClusters Computes the clusters from the input sce object and stores them back in sce object*

Description

seuratFindClusters Computes the clusters from the input sce object and stores them back in sce object

Usage

```
seuratFindClusters(
  inSCE,
  useAssay = "seuratScaledData",
  useReduction = c("pca", "ica"),
  dims = 10,
  algorithm = c("louvain", "multilevel", "SLM"),
  groupSingletons = TRUE,
  resolution = 0.8,
  externalReduction = NULL,
  verbose = TRUE
)
```

Arguments

inSCE	(sce) object from which clusters should be computed and stored in
useAssay	Assay containing scaled counts to use for clustering.
useReduction	Reduction method to use for computing clusters. One of "pca" or "ica". Default "pca".
dims	numeric value of how many components to use for computing clusters. Default 10.
algorithm	selected algorithm to compute clusters. One of "louvain", "multilevel", or "SLM". Use louvain for "original Louvain algorithm" and multilevel for "Louvain algorithm with multilevel refinement". Default louvain.
groupSingletons	boolean if singletons should be grouped together or not. Default TRUE.
resolution	Set the resolution parameter to find larger (value above 1) or smaller (value below 1) number of communities. Default 0.8.
externalReduction	Pass DimReduc object if PCA/ICA computed through other libraries. Default NULL.
verbose	Logical value indicating if informative messages should be displayed. Default is TRUE.

Value

Updated sce object which now contains the computed clusters

Examples

```
data(scExample, package = "singleCellTK")
## Not run:
sce <- seuratNormalizeData(sce, useAssay = "counts")
sce <- seuratFindHVG(sce, useAssay = "counts")
sce <- seuratScaleData(sce, useAssay = "counts")
sce <- seuratPCA(sce, useAssay = "counts")
sce <- seuratFindClusters(sce, useAssay = "counts")

## End(Not run)
```

`seuratFindHVG` *seuratFindHVG Find highly variable genes and store in the input sce object*

Description

seuratFindHVG Find highly variable genes and store in the input sce object

Usage

```
seuratFindHVG(
  inSCE,
  useAssay = "counts",
  hvgMethod = "vst",
  hvgNumber = 2000,
  altExp = FALSE,
  verbose = TRUE
)
```

Arguments

`inSCE`	(sce) object to compute highly variable genes from and to store back to it
`useAssay`	Specify the name of the assay to use for computation of variable genes. It is recommended to use a raw counts assay with the 'vst' method and normalized assay with all other methods. Default is `"counts"`.
`hvgMethod`	selected method to use for computation of highly variable genes. One of 'vst', 'dispersion', or 'mean.var.plot'. Default method is 'vst' which uses the raw counts. All other methods use normalized counts.
`hvgNumber`	numeric value of how many genes to select as highly variable. Default `2000`
`altExp`	Logical value indicating if the input object is an altExperiment. Default `FALSE`.
`verbose`	Logical value indicating if informative messages should be displayed. Default is `TRUE`.

Value

Updated SingleCellExperiment object with highly variable genes computation stored

Examples

```
data(scExample, package = "singleCellTK")
## Not run:
sce <- seuratNormalizeData(sce, useAssay = "counts")
sce <- seuratFindHVG(sce, useAssay = "counts")

## End(Not run)
```

---

seuratFindMarkers *seuratFindMarkers*

Description

seuratFindMarkers

Usage

```
seuratFindMarkers(
  inSCE,
  cells1 = NULL,
  cells2 = NULL,
  group1 = NULL,
  group2 = NULL,
  allGroup = NULL,
  conserved = FALSE,
  test = "wilcox",
  onlyPos = FALSE,
  minPCT = 0.1,
  threshUse = 0.25,
  verbose = TRUE
)
```

Arguments

inSCE	Input SingleCellExperiment object.
cells1	A list of sample names included in group1.
cells2	A list of sample names included in group2.
group1	Name of group1.
group2	Name of group2.
allGroup	Name of all groups.
conserved	Logical value indicating if markers conserved between two groups should be identified. Default is FALSE.

test	Test to use for DE. Default "wilcox".
onlyPos	Logical value indicating if only positive markers should be returned.
minPCT	Numeric value indicating the minimum fraction of min.pct cells in which genes are detected. Default is 0.1.
threshUse	Numeric value indicating the logFC threshold value on which on average, at least X-fold difference (log-scale) between the two groups of cells exists. Default is 0.25.
verbose	Logical value indicating if informative messages should be displayed. Default is TRUE.

Value

A SingleCellExperiment object that contains marker genes populated in a data.frame stored inside metadata slot.

---

seuratGenePlot *Compute and plot visualizations for marker genes*

Description

Compute and plot visualizations for marker genes

Usage

```
seuratGenePlot(
  inSCE,
  scaledAssayName = "seuratScaledData",
  plotType,
  features,
  groupVariable,
  splitBy = NULL,
  cols = c("lightgrey", "blue"),
  ncol = 1
)
```

Arguments

inSCE	Input SingleCellExperiment object.
scaledAssayName	Specify the name of the scaled assay stored in the input object.
plotType	Specify the type of the plot to compute. Options are limited to "ridge", "violing", "feature", "dot" and "heatmap".
features	Specify the features to compute the plot against.
groupVariable	Specify the column name from the colData slot that should be used as grouping variable.

splitBy	Specify the column name from the colData slot that should be used to split samples. Default is NULL.
cols	Specify two colors to form a gradient between. Default is c("lightgrey","blue").
ncol	Visualizations will be adjusted in "ncol" number of columns. Default is 1.

Value

Plot object

`seuratHeatmapPlot` *seuratHeatmapPlot Modifies the heatmap plot object so it contains specified number of heatmaps in a single plot*

Description

seuratHeatmapPlot Modifies the heatmap plot object so it contains specified number of heatmaps in a single plot

Usage

```
seuratHeatmapPlot(plotObject, dims, ncol, labels)
```

Arguments

plotObject	plot object computed from seuratComputeHeatmap() function
dims	numerical value of how many heatmaps to draw (default is 0)
ncol	numerical value indicating that in how many columns should the heatmaps be distrbuted (default is 2)
labels	list() of labels to draw on heatmaps

Value

modified plot object

seuratICA *seuratICA Computes ICA on the input sce object and stores the calculated independent components within the sce object*

Description

seuratICA Computes ICA on the input sce object and stores the calculated independent components within the sce object

Usage

```
seuratICA(
  inSCE,
  useAssay,
  reducedDimName = "seuratICA",
  features = NULL,
  nics = 20
)
```

Arguments

inSCE	(sce) object on which to compute ICA
useAssay	Assay containing scaled counts to use in ICA.
reducedDimName	Name of new reducedDims object containing Seurat ICA Default seuratICA.
features	Specify the feature names or rownames which should be used for computation of ICA. Default is NULL which will use the previously stored variable features.
nics	Number of independent components to compute. Default 20.

Value

Updated SingleCellExperiment object which now contains the computed independent components

Examples

```
data(scExample, package = "singleCellTK")
## Not run:
sce <- seuratNormalizeData(sce, useAssay = "counts")
sce <- seuratFindHVG(sce, useAssay = "counts")
sce <- seuratScaleData(sce, useAssay = "counts")
sce <- seuratICA(sce, useAssay = "counts")

## End(Not run)
```

seuratIntegration *seuratIntegration A wrapper function to Seurat Batch-Correction/Integration workflow.*

Description

seuratIntegration A wrapper function to Seurat Batch-Correction/Integration workflow.

Usage

```
seuratIntegration(
  inSCE,
  useAssay = "counts",
  batch,
  newAssayName = "SeuratIntegratedAssay",
  kAnchor,
  kFilter,
  kWeight,
  ndims = 10
)
```

Arguments

inSCE	Input SingleCellExperiment object that contains the assay to batch-correct.
useAssay	Assay to batch-correct.
batch	Batch variable from colData slot of SingleCellExperiment object.
newAssayName	Assay name for the batch-corrected output assay.
kAnchor	Number of neighbours to use for finding the anchors in the FindIntegrationAnchors function.
kFilter	Number of neighbours to use for filtering the anchors in the FindIntegrationAnchors function.
kWeight	Number of neighbours to use when weigthing the anchors in the IntegrateData function.
ndims	Number of dimensions to use. Default 10.

Value

A SingleCellExperiment object that contains the batch-corrected assay inside the altExp slot of the object

seuratJackStrawPlot *seuratJackStrawPlot Computes the plot object for jackstraw plot from the pca slot in the input sce object*

Description

seuratJackStrawPlot Computes the plot object for jackstraw plot from the pca slot in the input sce object

Usage

```
seuratJackStrawPlot(
  inSCE,
  dims = NULL,
  xmax = 0.1,
  ymax = 0.3,
  externalReduction = NULL
)
```

Arguments

inSCE	(sce) object from which to compute the jackstraw plot (pca should be computed)
dims	Number of components to plot in Jackstraw. If NULL, then all components are plotted Default NULL.
xmax	X-axis maximum on each QQ plot. Default 0.1.
ymax	Y-axis maximum on each QQ plot. Default 0.3.
externalReduction	Pass DimReduc object if PCA/ICA computed through other libraries. Default NULL.

Value

plot object

Examples

```
data(scExample, package = "singleCellTK")
## Not run:
sce <- seuratNormalizeData(sce, useAssay = "counts")
sce <- seuratFindHVG(sce, useAssay = "counts")
sce <- seuratScaleData(sce, useAssay = "counts")
sce <- seuratPCA(sce, useAssay = "counts")
sce <- seuratComputeJackStraw(sce, useAssay = "counts")
seuratJackStrawPlot(sce)

## End(Not run)
```

seuratNormalizeData *seuratNormalizeData Wrapper for NormalizeData() function from seurat library Normalizes the sce object according to the input parameters*

Description

seuratNormalizeData Wrapper for NormalizeData() function from seurat library Normalizes the sce object according to the input parameters

Usage

```
seuratNormalizeData(
  inSCE,
  useAssay,
  normAssayName = "seuratNormData",
  normalizationMethod = "LogNormalize",
  scaleFactor = 10000,
  verbose = TRUE
)
```

Arguments

inSCE	(sce) object to normalize
useAssay	Assay containing raw counts to use for normalization.
normAssayName	Name of new assay containing normalized data. Default seuratNormData.
normalizationMethod	selected normalization method. Default "LogNormalize".
scaleFactor	numeric value that represents the scaling factor. Default 10000.
verbose	Logical value indicating if informative messages should be displayed. Default is TRUE.

Value

Normalized SingleCellExperiment object

Examples

```
data(scExample, package = "singleCellTK")
## Not run:
sce <- seuratNormalizeData(sce, useAssay = "counts")

## End(Not run)
```

seuratPCA *seuratPCA Computes PCA on the input sce object and stores the calculated principal components within the sce object*

Description

seuratPCA Computes PCA on the input sce object and stores the calculated principal components within the sce object

Usage

```
seuratPCA(
  inSCE,
  useAssay = "seuratScaledData",
  reducedDimName = "seuratPCA",
  nPCs = 20,
  features = NULL,
  verbose = TRUE
)
```

Arguments

inSCE	(sce) object on which to compute PCA
useAssay	Assay containing scaled counts to use in PCA.
reducedDimName	Name of new reducedDims object containing Seurat PCA. Default seuratPCA.
nPCs	numeric value of how many components to compute. Default 20.
features	Specify the feature names or rownames which should be used for computation of PCA. Default is NULL which will use the previously stored variable features.
verbose	Logical value indicating if informative messages should be displayed. Default is TRUE.

Value

Updated SingleCellExperiment object which now contains the computed principal components

Examples

```
data(scExample, package = "singleCellTK")
## Not run:
sce <- seuratNormalizeData(sce, useAssay = "counts")
sce <- seuratFindHVG(sce, useAssay = "counts")
sce <- seuratScaleData(sce, useAssay = "counts")
sce <- seuratPCA(sce, useAssay = "counts")

## End(Not run)
```

`seuratPlotHVG` *seuratPlotHVG Plot highly variable genes from input sce object (must have highly variable genes computations stored)*

Description

seuratPlotHVG Plot highly variable genes from input sce object (must have highly variable genes computations stored)

Usage

```
seuratPlotHVG(inSCE, labelPoints = 0)
```

Arguments

`inSCE`	(sce) object that contains the highly variable genes computations
`labelPoints`	Numeric value indicating the number of top genes that should be labeled. Default is `0`, which will not label any point.

Value

plot object

Examples

```
data(scExample, package = "singleCellTK")
## Not run:
sce <- seuratNormalizeData(sce, useAssay = "counts")
sce <- seuratFindHVG(sce, useAssay = "counts")
seuratPlotHVG(sce)

## End(Not run)
```

`seuratReductionPlot` *seuratReductionPlot Plots the selected dimensionality reduction method*

Description

seuratReductionPlot Plots the selected dimensionality reduction method

Usage

```
seuratReductionPlot(
  inSCE,
  useReduction = c("pca", "ica", "tsne", "umap"),
  showLegend = FALSE,
  groupBy = NULL,
  splitBy = NULL
)
```

Arguments

inSCE	(sce) object which has the selected dimensionality reduction algorithm already computed and stored
useReduction	Dimentionality reduction to plot. One of "pca", "ica", "tsne", or "umap". Default "umap".
showLegend	Select if legends and labels should be shown on the output plot or not. Either "TRUE" or "FALSE". Default FALSE.
groupBy	Specify a colData column name that be used for grouping. Default is NULL.
splitBy	Specify a colData column name that be used for splitting the output plot. Default is NULL.

Value

plot object

Examples

```
data(scExample, package = "singleCellTK")
## Not run:
sce <- seuratNormalizeData(sce, useAssay = "counts")
sce <- seuratFindHVG(sce, useAssay = "counts")
sce <- seuratScaleData(sce, useAssay = "counts")
sce <- seuratPCA(sce, useAssay = "counts")
seuratReductionPlot(sce, useReductionPlot = "pca")
## End(Not run)
```

---

seuratReport *Computes an HTML report from the Seurat workflow and returns the output SCE object with the computations stored in it.*

Description

Computes an HTML report from the Seurat workflow and returns the output SCE object with the computations stored in it.

Usage

```
seuratReport(
  inSCE,
  outputFile = NULL,
  outputDir = NULL,
  subtitle = "BUMC Single Cell Sequencing Core",
  authors = "Tianmu (Timo) Hu, Irzam Sarfraz",
  sce = NULL,
  biological.group = NULL,
  phenotype.groups = NULL,
  selected.markers = NULL,
  clustering.resolution = 0.8,
  variable.features = 2000,
  pc.count = 10,
  showSession = TRUE,
  pdf = TRUE
)
```

Arguments

inSCE	Input SingleCellExperiment object.
outputFile	Specify the name of the generated output HTML file. If NULL then the output file name will be based on the name of the Rmarkdown template. Default NULL.
outputDir	Specify the name of the output directory to save the rendered HTML file. If NULL the file is stored to the current working directory.
subtitle	A character value specifying the subtitle to use in the Seurat report.
authors	A character value specifying the names of the authors to use in the Seurat report.
sce	A character value specifying the path of the input SingleCellExperiment object.
biological.group	A character value that specifies the name of the colData column to use as the main biological group in the seurat report for differential expression and grouping.
phenotype.groups	A character vector that specifies the names of the colData columns to use for differential expression in addition to the biological.group parameter.
selected.markers	A character vector specifying the user decided gene symbols of pre-selected markers that be used to generate gene plots in addition to the gene markers computed from differential expression.
clustering.resolution	A numeric value indicating the resolution to use with clustering. Default is 0.8.
variable.features	A numeric value indicating the number of top variable genes to identify in the seurat report. Default is 2000.

pc.count	A numeric value indicating the number of principal components to use in the analysis workflow. Default is 10.
showSession	A logical value indicating if session information should be displayed or not. Default is TRUE.
pdf	A logical value indicating if a pdf should also be generated for each figure in the report. Default is TRUE.

Value

A SingleCellExperiment object that has the seurat computations stored and can be used to interactively visualize the plots by importing in the singleCellTK user interface.

---

seuratRunTSNE *seuratRunTSNE Computes tSNE from the given sce object and stores the tSNE computations back into the sce object*

Description

seuratRunTSNE Computes tSNE from the given sce object and stores the tSNE computations back into the sce object

Usage

```
seuratRunTSNE(
  inSCE,
  useReduction = c("pca", "ica"),
  reducedDimName = "seuratTSNE",
  dims = 10,
  perplexity = 30,
  externalReduction = NULL
)
```

Arguments

inSCE	(sce) object on which to compute the tSNE
useReduction	selected reduction algorithm to use for computing tSNE. One of "pca" or "ica". Default "pca".
reducedDimName	Name of new reducedDims object containing Seurat tSNE Default seuratTSNE.
dims	Number of reduction components to use for tSNE computation. Default 10.
perplexity	Adjust the perplexity tuneable parameter for the underlying tSNE call. Default 30.
externalReduction	Pass DimReduc object if PCA/ICA computed through other libraries. Default NULL.

Value

Updated sce object with tSNE computations stored

seuratRunUMAP *seuratRunUMAP Computes UMAP from the given sce object and stores the UMAP computations back into the sce object*

Description

seuratRunUMAP Computes UMAP from the given sce object and stores the UMAP computations back into the sce object

Usage

```
seuratRunUMAP(
  inSCE,
  useReduction = c("pca", "ica"),
  reducedDimName = "seuratUMAP",
  dims = 10,
  minDist = 0.3,
  nNeighbors = 30L,
  spread = 1,
  externalReduction = NULL,
  verbose = TRUE
)
```

Arguments

inSCE	(sce) object on which to compute the UMAP
useReduction	Reduction to use for computing UMAP. One of "pca" or "ica". Default is "pca".
reducedDimName	Name of new reducedDims object containing Seurat UMAP Default seuratUMAP.
dims	Numerical value of how many reduction components to use for UMAP computation. Default 10.
minDist	Sets the "min.dist" parameter to the underlying UMAP call. See RunUMAP for more information. Default 0.3.
nNeighbors	Sets the "n.neighbors" parameter to the underlying UMAP call. See RunUMAP for more information. Default 30L.
spread	Sets the "spread" parameter to the underlying UMAP call. See RunUMAP for more information. Default 1.
externalReduction	Pass DimReduc object if PCA/ICA computed through other libraries. Default NULL.
verbose	Logical value indicating if informative messages should be displayed. Default is TRUE.

Value

Updated sce object with UMAP computations stored

Examples

```
data(scExample, package = "singleCellTK")
## Not run:
sce <- seuratNormalizeData(sce, useAssay = "counts")
sce <- seuratFindHVG(sce, useAssay = "counts")
sce <- seuratScaleData(sce, useAssay = "counts")
sce <- seuratPCA(sce, useAssay = "counts")
sce <- seuratFindClusters(sce, useAssay = "counts")
sce <- seuratRunUMAP(sce, useReduction = "pca")

## End(Not run)
```

`seuratScaleData` *seuratScaleData Scales the input sce object according to the input parameters*

Description

seuratScaleData Scales the input sce object according to the input parameters

Usage

```
seuratScaleData(
  inSCE,
  useAssay = "seuratNormData",
  scaledAssayName = "seuratScaledData",
  model = "linear",
  scale = TRUE,
  center = TRUE,
  scaleMax = 10,
  verbose = TRUE
)
```

Arguments

`inSCE`	(sce) object to scale
`useAssay`	Assay containing normalized counts to scale.
`scaledAssayName`	Name of new assay containing scaled data. Default `seuratScaledData`.
`model`	selected model to use for scaling data. Default `"linear"`.
`scale`	boolean if data should be scaled or not. Default `TRUE`.
`center`	boolean if data should be centered or not. Default `TRUE`
`scaleMax`	maximum numeric value to return for scaled data. Default `10`.
`verbose`	Logical value indicating if informative messages should be displayed. Default is `TRUE`.

Value

Scaled SingleCellExperiment object

Examples

```
data(scExample, package = "singleCellTK")
## Not run:
sce <- seuratNormalizeData(sce, useAssay = "counts")
sce <- seuratFindHVG(sce, useAssay = "counts")
sce <- seuratScaleData(sce, useAssay = "counts")

## End(Not run)
```

---

seuratSCTransform *seuratSCTransform Runs the SCTransform function to transform/normalize the input data*

Description

seuratSCTransform Runs the SCTransform function to transform/normalize the input data

Usage

```
seuratSCTransform(
  inSCE,
  normAssayName = "SCTCounts",
  useAssay = "counts",
  verbose = TRUE
)
```

Arguments

inSCE	Input SingleCellExperiment object
normAssayName	Name for the output data assay. Default "SCTCounts".
useAssay	Name for the input data assay. Default "counts".
verbose	Logical value indicating if informative messages should be displayed. Default is TRUE.

Value

Updated SingleCellExperiment object containing the transformed data

Examples

```
data("mouseBrainSubsetSCE", package = "singleCellTK")
mouseBrainSubsetSCE <- seuratSCTransform(mouseBrainSubsetSCE)
```

seuratVariableFeatures *Get variable feature names after running seuratFindHVG function*

Description

Get variable feature names after running seuratFindHVG function

Usage

```
seuratVariableFeatures(inSCE)
```

Arguments

inSCE	Input SingleCellExperiment object.

Value

A list of variable feature names.

simpleLog *A decorator that prints the arguments to the decorated function*

Description

A decorator that prints the arguments to the decorated function

Usage

```
simpleLog(f)
```

Arguments

f	A function to decorate

Value

Prints message

`singleCellTK` *Run the single cell analysis app*

Description

Use this function to run the single cell analysis app.

Usage

```
singleCellTK(inSCE = NULL, includeVersion = TRUE, theme = "yeti")
```

Arguments

`inSCE`	Input SingleCellExperiment object.
`includeVersion`	Include the version number in the SCTK header. The default is TRUE.
`theme`	The bootswatch theme to use for the singleCellTK UI. The default is 'flatly'.

Value

The shiny app will open

Examples

```
## Not run:
#Upload data through the app
singleCellTK()

# Load the app with a SingleCellExperiment object
data("mouseBrainSubsetSCE")
singleCellTK(mouseBrainSubsetSCE)

## End(Not run)
```

`subDiffEx` *Passes the output of generateSimulatedData() to differential expression tests, picking either t-tests or ANOVA for data with only two conditions or multiple conditions, respectively.*

Description

Passes the output of generateSimulatedData() to differential expression tests, picking either t-tests or ANOVA for data with only two conditions or multiple conditions, respectively.

Usage

```
subDiffEx(tempData)

subDiffExttest(countMatrix, class.labels, test.type = "t.equalvar")

subDiffExANOVA(countMatrix, condition)
```

Arguments

tempData	Matrix. The output of generateSimulatedData(), where the first row contains condition labels.
countMatrix	Matrix. A simulated counts matrix, sans labels.
class.labels	Factor. The condition labels for the simulated cells. Will be coerced into 1's and 0's.
test.type	Type of test to perform. The default is t.equalvar.
condition	Factor. The condition labels for the simulated cells.

Value

subDiffEx(): A vector of fdr-adjusted p-values for all genes. Nonviable results (such as for genes with 0 counts in a simulated dataset) are coerced to 1.

subDiffExttest(): A vector of fdr-adjusted p-values for all genes. Nonviable results (such as for genes with 0 counts in a simulated dataset) are coerced to 1.

subDiffExANOVA(): A vector of fdr-adjusted p-values for all genes. Nonviable results (such as for genes with 0 counts in a simulated dataset) are coerced to 1.

Functions

- subDiffEx:
- subDiffExttest: Runs t-tests on all genes in a simulated dataset with 2 conditions, and adjusts for FDR.
- subDiffExANOVA: Runs ANOVA on all genes in a simulated dataset with more than 2 conditions, and adjusts for FDR.

Examples

```
data("mouseBrainSubsetSCE")
res <- generateSimulatedData(
         totalReads = 1000, cells=10,
         originalData = assay(mouseBrainSubsetSCE, "counts"),
         realLabels = colData(mouseBrainSubsetSCE)[, "level1class"])
tempSigDiff <- subDiffEx(res)

data("mouseBrainSubsetSCE")
#sort first 100 expressed genes
ord <- rownames(mouseBrainSubsetSCE)[
  order(rowSums(assay(mouseBrainSubsetSCE, "counts")),
```

```
        decreasing = TRUE)][seq(100)]
#subset to those first 100 genes
subset <- mouseBrainSubsetSCE[ord, ]
res <- generateSimulatedData(totalReads = 1000, cells=10,
                             originalData = assay(subset, "counts"),
                             realLabels = colData(subset)[, "level1class"])
realLabels <- res[1, ]
output <- res[-1, ]
fdr <- subDiffExttest(output, realLabels)

data("mouseBrainSubsetSCE")
#sort first 100 expressed genes
ord <- rownames(mouseBrainSubsetSCE)[
  order(rowSums(assay(mouseBrainSubsetSCE, "counts")),
        decreasing = TRUE)][seq(100)]
# subset to those first 100 genes
subset <- mouseBrainSubsetSCE[ord, ]
res <- generateSimulatedData(totalReads = 1000, cells=10,
                             originalData = assay(subset, "counts"),
                             realLabels = colData(subset)[, "level2class"])
realLabels <- res[1, ]
output <- res[-1, ]
fdr <- subDiffExANOVA(output, realLabels)
```

`subsetSCECols` *Subset a SingleCellExperiment object by columns*

Description

Used to peform subsetting of a SingleCellExperiment object using a variety of methods that indicate the correct columns to keep. The various methods, `index`, `bool`, and `colData`, can be used in conjunction with one another.

Usage

```
subsetSCECols(inSCE, index = NULL, bool = NULL, colData = NULL)
```

Arguments

`inSCE`	Input SingleCellExperiment object.
`index`	Integer vector. Vector of indicies indicating which columns to keep. If `NULL`, this will not be used for subsetting. Default `NULL`.
`bool`	Boolean vector. Vector of `TRUE` or `FALSE` indicating which columns should be kept. Needs to be the same length as the number of columns in `inSCE`. If `NULL`, this will not be used for subsetting. Default `NULL`.

colData — Character. An expression that will identify a subset of columns using variables found in the colData of inSCE. For example, if x is a numeric vector in colData, then "x < 5" will return all columns with x less than 5. Single quotes should be used for character strings. For example, "y == 'yes'" will return all columns where y is "yes". Multiple expressions can be evaluated by placing them in a vector. For example c("x < 5","y =='yes'") will apply both operations for subsetting. If NULL, this will not be used for subsetting. Default NULL.

Value

A SingleCellExperiment object that has been subsetted by colData.

Author(s)

Joshua D. Campbell

Examples

```
data(scExample)
sce <- subsetSCECols(sce, colData = "type != 'EmptyDroplet'")
```

---

subsetSCERows *Subset a SingleCellExperiment object by rows*

Description

Used to peform subsetting of a SingleCellExperiment object using a variety of methods that indicate the correct rows to keep. The various methods, index, bool, and rowData, can be used in conjunction with one another. If returnAsAltExp is set to TRUE, then the returned object will have the same number of rows as the input inSCE as the subsetted object will be stored in the altExp slot.

Usage

```
subsetSCERows(
  inSCE,
  index = NULL,
  bool = NULL,
  rowData = NULL,
  returnAsAltExp = TRUE,
  altExpName = "subset",
  prependAltExpName = TRUE
)
```

Arguments

inSCE	Input SingleCellExperiment object.
index	Integer vector. Vector of indicies indicating which rows to keep. If NULL, this will not be used for subsetting. Default NULL.
bool	Boolean vector. Vector of TRUE or FALSE indicating which rows should be kept. Needs to be the same length as the number of rows in inSCE. If NULL, this will not be used for subsetting. Default NULL.
rowData	Character. An expression that will identify a subset of rows using variables found in the rowData of inSCE. For example, if x is a numeric vector in rowData, then "x < 5" will return all rows with x less than 5. Single quotes should be used for character strings. For example, "y == 'yes'" will return all rows where y is "yes". Multiple expressions can be evaluated by placing them in a vector. For example c("x < 5","y =='yes'") will apply both operations for subsetting. If NULL, this will not be used for subsetting. Default NULL.
returnAsAltExp	Boolean. If TRUE, the subsetted SingleCellExperiment object will be returned in the altExp slot of inSCE. If FALSE, the subsetted SingleCellExperiment object will be directly returned.
altExpName	Character. Name of the alternative experiment object to add if returnAsAltExp = TRUE. Default subset.
prependAltExpName	Boolean. If TRUE, altExpName will be added to the beginning of the assay names in the altExp object. This is only utilized if returnAsAltExp = TRUE. Default TRUE.

Value

A SingleCellExperiment object that has been subsetted by rowData.

Author(s)

Joshua D. Campbell

Examples

```
data(scExample)

# Set a variable up in the rowData indicating mitochondrial genes
rowData(sce)$isMito <- ifelse(grepl("^MT-", rowData(sce)$feature_name),
                              "yes", "no")
sce <- subsetSCERows(sce, rowData = "isMito == 'yes'")
```

summarizeSCE *Summarize an assay in a SingleCellExperiment*

Description

Creates a table of summary metrics from an input SingleCellExperiment

Usage

```
summarizeSCE(inSCE, useAssay = NULL, sampleVariableName = NULL)
```

Arguments

inSCE	Input SingleCellExperiment object.
useAssay	Indicate which assay to summarize. If NULL, then the first assay in inSCE will be used. Default NULL.
sampleVariableName	Variable name in colData denoting which sample each cell belongs to. If NULL, all cells will be assumed to come from the same sample. Default "sample".

Value

A data.frame object of summary metrics.

Examples

```
data("mouseBrainSubsetSCE")
summarizeSCE(mouseBrainSubsetSCE, sample = NULL)
```

trimCounts *Trim Counts*

Description

Trims an input count matrix such that each value greater than a threshold value and each value less than a provided lower threshold value is trimmed to the lower treshold value.

Usage

```
trimCounts(counts, trimValue = c(10, -10))
```

Arguments

counts	matrix
trimValue	where trimValue[1] for upper threshold and trimValue[2] as lower threshold. Default is c(10,-10)

Value

trimmed counts matrix

Examples

```
data(sce_chcl, package = "scds")
assay(sce_chcl, "countsTrimmed") <- trimCounts(assay(sce_chcl, "counts"),
                                               c(10, -10))
```

Index