Package 'singleCellTK'

October 16, 2019

Type Package

Title Interactive Analysis of Single Cell RNA-Seq Data

Version 1.4.2

Author David Jenkins

Maintainer David Jenkins <dfj@bu.edu>

Depends R (>= 3.5), SummarizedExperiment, SingleCellExperiment, DelayedArray, Biobase

Description Run common single cell analysis directly through your browser including differential expression, downsampling analysis, and clustering.

License MIT + file LICENSE

Encoding UTF-8

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

LazyData TRUE

Imports ape, colourpicker, cluster, ComplexHeatmap, data.table, DESeq2, DT, ggplot2, ggtree, gridExtra, GSVA (>= 1.26.0), GSVAdata, limma, MAST, matrixStats, methods, multtest, plotly, RColorBrewer, Rtsne, S4Vectors, shiny, shinyjs, sva, reshape2, AnnotationDbi, shinyalert, circlize, enrichR, shinycssloaders, shinythemes, umap

RoxygenNote 6.1.1

Suggests testthat, Rsubread, BiocStyle, knitr, bladderbatch, rmarkdown, org.Mm.eg.db, org.Hs.eg.db, scRNAseq, xtable, spelling, GSEABase, lintr

VignetteBuilder knitr

URL https://compbiomed.github.io/sctk_docs/

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

Language en-US

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

git_branch RELEASE_3_9

git_last_commit 0285e0c

git_last_commit_date 2019-08-04

Date/Publication 2019-10-15

R topics documented:

`alignSingleCellData` *Align Single Cell RNA-Seq Data and Create a SCtkExperiment Object*

Description

Align Single Cell RNA-Seq Data and Create a SCtkExperiment Object

Usage

```
alignSingleCellData(inputfile1, inputfile2 = NULL, indexPath,
  gtfAnnotation, outputDir = NULL, sampleAnnotations = NULL,
  featureAnnotations = NULL, threads = 1, saveBam = FALSE,
  saveCountFiles = FALSE, isPairedEnd = FALSE)
```

Arguments

inputfile1	An input file or list of files. Files can be fastq, fastq.gz, or bam, but must all be of the same type. Sample names will be the full file name, without _1.fastq.gz, .fastq.gz, _1.fastq, .fastq or .bam endings.
inputfile2	If fastq files are provided in input list, a list of corresponding paired fastq files, if applicable.
indexPath	Path to the Rsubread genome index.
gtfAnnotation	Path to the GTF gene annotation to use. This must correspond to the genome specified in indexPath.
outputDir	If saveBam or saveCountFiles is TRUE, specify a directory in which to save the output files.
sampleAnnotations	A data.frame of sample annotations, with samples as rows and annotations in columns. The sample names must be identical to and in the same order as the list of files in inputfile1. Alignment statistics will be added to the annotation data frame.
featureAnnotations	An optional data.frame of probe annotations, with probes as rows and probe annotations in columns.
threads	Number of threads to use during alignment. The default is 1.
saveBam	If TRUE, bam alignment files will be saved in the outputDir. The default is FALSE.
saveCountFiles	If TRUE, per sample gene count files will be saved in the outputDir. The default is FALSE.
isPairedEnd	If input files are .bam, indicate whether the input bam files are paired end.

Value

Object to import into the shiny app.

Examples

```
## Not run:
singlecellobject <- alignSingleCellData(
  inputfile1 = c("/path/to/sample1_1.fastq.gz",
                 "/path/to/sample2_1.fastq.gz"),
  inputfile2 = c("/path/to/sample1_2.fastq.gz",
                 "/path/to/sample2_2.fastq.gz"),
  indexPath = "/path/to/genome/index",
  gtfAnnotation = "/path/to/gene/annotations.gtf",
  sampleAnnotations = sample.annotation.df,
  threads=4)
## End(Not run)
```

`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"])
```

`ComBatSCE` *ComBatSCE*

Description

Run ComBat on a SCtkExperiment object

Usage

```
ComBatSCE(inSCE, batch, useAssay = "logcounts",
  par.prior = "Parametric", covariates = NULL, mean.only = FALSE,
  ref.batch = NULL)
```

Arguments

inSCE	Input SCtkExperiment object. Required
batch	The name of a column in colData to use as the batch variable. Required
useAssay	The assay to use for ComBat. The default is "logcounts"
par.prior	TRUE indicates parametric adjustments will be used, FALSE indicates non-parametric adjustments will be used. Accepted parameters: "Parametric" or "Non-parametric"
covariates	List of other column names in colData to be added to the ComBat model as covariates
mean.only	If TRUE ComBat only corrects the mean of the batch effect
ref.batch	If given, will use the selected batch as a reference for batch adjustment.

Value

ComBat matrix based on inputs. You can save this matrix into the SCtkExperiment with assay()

Examples

```
if(requireNamespace("bladderbatch", quietly = TRUE)) {
  library(bladderbatch)
  data(bladderdata)

  #subset for testing
  dat <- bladderEset[1:50,]
  dat <- as(as(dat, "SummarizedExperiment"), "SCtkExperiment")
  mod <- stats::model.matrix(~as.factor(cancer), data = colData(dat))

  # parametric adjustment
  combat_edata1 <- ComBatSCE(inSCE = dat, useAssay = "exprs",
                             batch = "batch", covariates = NULL)
  assay(dat, "parametric_combat") <- combat_edata1

  # non-parametric adjustment, mean-only version
  combat_edata2 <- ComBatSCE(inSCE = dat, useAssay = "exprs",
                             batch = "batch", par.prior = "Non-parametric",
                             mean.only = TRUE, covariates = NULL)
  assay(dat, "nonparametric_combat_meanonly") <- combat_edata2

  # reference-batch version, with covariates
  combat_edata3 <- ComBatSCE(inSCE = dat, useAssay = "exprs",
                             batch = "batch", covariates = "cancer",
                             ref.batch = 3)
  assay(dat, "refbatch_combat_wcov") <- combat_edata3
  assays(dat)
}
```

convertGeneIDs *Convert Gene IDs*

Description

Convert the gene IDs in a SingleCellExperiment object using Bioconductor org.*.eg.db data packages. Because annotation databases do not have a 1:1 relationship, this tool removes rows with no corresponding annotation in your desired annotation, and remove any duplicate annotations after conversion.

Usage

```
convertGeneIDs(inSCE, inSymbol, outSymbol, database = "org.Hs.eg.db")
```

Arguments

inSCE	Input SCtkExperiment object. Required
inSymbol	The input symbol type
outSymbol	The output symbol type
database	The org.*.eg.db database to use. The default is org.Hs.eg.db

Value

A SCtkExperiment with converted gene IDs.

Examples

```
if(requireNamespace("org.Mm.eg.db", quietly = TRUE)) {
  #convert mouse gene symbols to ensembl IDs
  library("org.Mm.eg.db")
  sample(rownames(mouseBrainSubsetSCE), 50)
  mouseBrainSubsetSymbol <- convertGeneIDs(inSCE = mouseBrainSubsetSCE,
                                           inSymbol = "SYMBOL",
                                           outSymbol = "ENSEMBL",
                                           database = "org.Mm.eg.db")
  sample(rownames(mouseBrainSubsetSymbol), 50)
}
```

createSCE *Create a SCtkExperiment object*

Description

From a file of counts and a file of annotation information, create a SCtkExperiment object.

Usage

```
createSCE(assayFile = NULL, annotFile = NULL, featureFile = NULL,
  assayName = "counts", inputDataFrames = FALSE,
  createLogCounts = TRUE)
```

Arguments

assayFile	The path to a text file that contains a header row of sample names, and rows of raw counts per gene for those samples.
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.
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.
createLogCounts	If TRUE, create a log2(counts+1) normalized assay and include it in the object. The default is TRUE

Value

a SCtkExperiment object

Examples

```
data("mouseBrainSubsetSCE")
counts_mat <- assay(mouseBrainSubsetSCE, "counts")
sample_annot <- colData(mouseBrainSubsetSCE)
row_annot <- rowData(mouseBrainSubsetSCE)
newSCE <- createSCE(assayFile = counts_mat, annotFile = sample_annot,
                    featureFile = row_annot, assayName = "counts",
                    inputDataFrames = TRUE, createLogCounts = TRUE)
```

---

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
: SCtkExperiment. The SCtkExperiment 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[1:1000,]
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`	SCtkExperiment. The SCtkExperiment 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[1:1000,]
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 SCtkExperiment object. Required
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

```
## Not run:
enrichRSCE(mouseBrainSubsetSCE, "Cmtm5", "GO_Cellular_Component_2017")

## End(Not run)
```

`filterSCData` *Filter Genes and Samples from a Single Cell Object*

Description

Filter Genes and Samples from a Single Cell Object

Usage

```
filterSCData(inSCE, useAssay = "counts", deletesamples = NULL,
  removeNoExpress = TRUE, removeBottom = 0.5,
  minimumDetectGenes = 1700, filterSpike = TRUE)
```

Arguments

`inSCE`	Input SCtkExperiment object. Required
`useAssay`	Indicate which assay to use for filtering. Default is "counts"
`deletesamples`	List of samples to delete from the object.
`removeNoExpress`	Remove genes that have no expression across all samples. The default is true
`removeBottom`	Fraction of low expression genes to remove from the single cell object. This occurs after removeNoExpress. The default is 0.50.
`minimumDetectGenes`	Minimum number of genes with at least 1 count to include a sample in the single cell object. The default is 1700.
`filterSpike`	Apply filtering to Spike in controls (indicated by isSpike). The default is TRUE.

Value

The filtered single cell object.

Examples

```
data("mouseBrainSubsetSCE")
mouseBrainSubsetSCE <- filterSCData(mouseBrainSubsetSCE,
                                    deletesamples="X1772063061_G11")
```

`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 SCtkExperiment object, return the binary or continuous expression of the genes.*

Description

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

Usage

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

Arguments

inSCE	Input SCtkExperiment object. Required
gene	gene list
binary	"Binary" for binary expression or "Continuous" for a gradient. Default: "Binary"
useAssay	Indicate which assay to use. The default is "counts".

Value

getBiomarker(): A data.frame of expression values

Examples

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

getClusterInputData *Get data to use as input clustering algorithms*

Description

Get data to use as input clustering algorithms

Usage

```
getClusterInputData(inSCE, inputData, useAssay = "logcounts",
  reducedDimName = NULL)
```

Arguments

inSCE	Input SCtkExperiment object. Required
inputData	A string ("Raw Data", "PCA Components", "tSNE Components", "UMAP Components")
useAssay	Indicate which assay to use for PCA. Default is "logcounts"
reducedDimName	If clustering on PCA, t-SNE or UMAP data, dimension name. The toolkit will store data with the pattern <ASSAY>_<ALGORITHM>.

Value

Cluster input data

Examples

```
data("mouseBrainSubsetSCE")
getClusterInputData(mouseBrainSubsetSCE, "PCA Components",
                    useAssay = "logcounts", reducedDimName = "PCA_logcounts")
```

getPCA *Get and plot PCA components for a SCtkE object*

Description

Selects the 500 most variable genes in the SCE, performs PCA based on them and stores the values in the reducedDims slot of the SCE object.

Usage

```
getPCA(inSCE, useAssay = "logcounts", reducedDimName = "PCA")
```

Arguments

inSCE	Input SCtkExperiment object. Required
useAssay	Indicate which assay to use for PCA. Default is "counts"
reducedDimName	Store the PCA data with this name. The default is PCA. The toolkit will store data with the pattern <ASSAY>_<ALGORITHM>.

Value

A SCtkE object with the specified reducedDim and pcaVariances updated

Examples

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

---

getTSNE *Run t-SNE dimensionality reduction method on the assay data.*

Description

Run t-SNE dimensionality reduction method on the assay data.

Usage

```
getTSNE(inSCE, useAssay = "logcounts", reducedDimName = "TSNE",
  n_iterations = 1000, perplexity = NULL)
```

Arguments

inSCE	Input SCtkExperiment object. Required
useAssay	Indicate which assay to use. The default is "logcounts".
reducedDimName	a name to store the results of the dimension reductions
n_iterations	maximum iterations. Default is 1000
perplexity	perplexity parameter. Default is 5

Value

A SCtkE object with the specified reducedDim and pcaVariances updated

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")
reducedDims(mouseBrainSubsetSCE)
```

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 = "logcounts", reducedDimName = "UMAP",
  n_neighbors = 5, n_iterations = 200, alpha = 1)
```

Arguments

inSCE	Input SCtkExperiment object. Required
useAssay	Indicate which assay to use. The default is "logcounts".
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.
n_neighbors	specify the number of nearest neighbors. Default is 5.
n_iterations	number of iterations performed during layout optimization. Default is 200.
alpha	initial value of "learning rate" of layout optimization. Default is 1.

Value

a SCtkExperiment object with the reduced dimensions updated under reducedDimName specified.

Examples

```
umap_res <- getUMAP(inSCE = mouseBrainSubsetSCE, useAssay = "counts",
                    reducedDimName = "UMAP", n_neighbors = 3, n_iterations = 200,
                    alpha = 1)
reducedDims(umap_res)
```

gsvaSCE *Run GSVA analysis on a SCtkExperiment object.*

Description

Run GSVA analysis on a SCtkExperiment object.

Usage

```
gsvaSCE(inSCE, useAssay = "logcounts", pathwaySource, pathwayNames, ...)

gsvaPlot(inSCE, gsvaData, plotType, condition = NULL,
  show_column_names = TRUE, show_row_names = TRUE, text_size = 12)
```

Arguments

inSCE	Input SCtkExperiment object. Required
useAssay	Indicate which assay to use. The default is "logcounts"
pathwaySource	The pathway source if "Manual Input", the pathwayNames should be rowData annotations that are (0,1) vectors. If, "MSigDB c2 (Human, Entrez ID only)", the pathwayNames should be pathways from MSigDB c2 or "ALL" to run on all available pathways.
pathwayNames	List of pathway names to run, depending on pathwaySource parameter.
...	Parameters to pass to gsva()
gsvaData	GSVA data to plot. Required.
plotType	The type of plot to use, "Violin" or "Heatmap". Required.
condition	The condition(s) to use for the Violin plot, or the condition(s) to add as color bars above the Heatmap. Required for Violin, optional for Heatmap.
show_column_names	Display the column labels on the heatmap. The default is TRUE
show_row_names	Display the row labels on the heatmap. The default is TRUE.
text_size	Text size for plots. The default is 12

Value

gsvaSCE(): A data.frame of pathway activity scores from GSVA.

gsvaPlot(): The requested plot of the GSVA results.

Functions

- gsvaPlot: Plot GSVA results.
  Plot GSVA Results

Examples

```
utils::data(maits, package = "MAST")
utils::data(c2BroadSets, package = "GSVAdata")
maitslogtpm <- t(maits$expressionmat)
genesToSubset <- rownames(maitslogtpm)[which(rownames(maitslogtpm) %in%
                GSEABase::geneIds(c2BroadSets[["KEGG_PROTEASOME"]]))]
maitslogtpm <- maitslogtpm[rownames(maitslogtpm) %in% genesToSubset, ]
maitsfeatures <- maits$fdat[rownames(maits$fdat) %in% genesToSubset, ]
maitsSCE <- createSCE(assayFile = maitslogtpm, annotFile = maits$cdat,
                      featureFile = maitsfeatures, assayName = "logtpm",
                      inputDataFrames = TRUE, createLogCounts = FALSE)
rowData(maitsSCE)$testbiomarker <- rep(1, nrow(maitsSCE))
res <- gsvaSCE(inSCE = maitsSCE, useAssay = "logtpm",
               pathwaySource = "Manual Input", pathwayNames = "testbiomarker",
               parallel.sz = 1)
#Create a small example to run
utils::data(maits, package = "MAST")
utils::data(c2BroadSets, package = "GSVAdata")
maitslogtpm <- t(maits$expressionmat)
genesToSubset <- rownames(maitslogtpm)[which(rownames(maitslogtpm) %in%
                GSEABase::geneIds(c2BroadSets[["KEGG_PROTEASOME"]]))]
maitslogtpm <- maitslogtpm[rownames(maitslogtpm) %in% genesToSubset, ]
```

```
maitsfeatures <- maits$fdat[rownames(maits$fdat) %in% genesToSubset, ]
maitsSCE <- createSCE(assayFile = maitslogtpm, annotFile = maits$cdat,
                      featureFile = maitsfeatures, assayName = "logtpm",
                      inputDataFrames = TRUE, createLogCounts = FALSE)
rowData(maitsSCE)$testbiomarker <- rep(1, nrow(maitsSCE))
res <- gsvaSCE(inSCE = maitsSCE, useAssay = "logtpm",
               pathwaySource = "Manual Input", pathwayNames = "testbiomarker",
               parallel.sz = 1)
gsvaPlot(inSCE = maitsSCE, gsvaData = res,
         plotType = "Violin", condition = "condition")
```

`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`	SCtkExperiment. The SCtkExperiment 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)
```

MAST *MAST*

Description

Run and visualize MAST analysis on a SCtkExperiment object.

Usage

```
MAST(inSCE, condition = NULL, interest.level = NULL,
  freqExpressed = 0.1, fcThreshold = log2(1.5), p.value = 0.05,
  useThresh = FALSE, useAssay = "logcounts")

thresholdGenes(inSCE, useAssay = "logcounts")

MASTviolin(inSCE, useAssay = "logcounts", fcHurdleSig, samplesize = 49,
  threshP = FALSE, condition)

MASTregression(inSCE, useAssay = "logcounts", fcHurdleSig,
  samplesize = 49, threshP = FALSE, condition)
```

Arguments

inSCE	Input SCtkExperiment object. Required
condition	select variable (from the colData) that is used for the model.
interest.level	If the condition of interest has more than two factors, indicate which level should be used to compare to all other samples.
freqExpressed	Filter genes that are expressed in at least this fraction of cells. The default is expression in 0.1 of samples.
fcThreshold	Minimum fold change for differentially expressed gene.
p.value	p values for selecting the hurdle result, default is 0.05
useThresh	Use adaptive thresholding to filter genes. The default is FALSE.
useAssay	The assay to use for the MAST calculations. The default is "logcounts"
fcHurdleSig	The filtered result from hurdle model
samplesize	The number of most significant genes
threshP	Plot threshold values from adaptive thresholding. Default is FALSE

Value

MAST(): A data.frame of differentially expressed genes with p-values.

thresholdGenes(): list of thresholded counts (on natural scale), thresholds, bins, densities estimated on each bin, and the original data from MAST::thresholdSCRNACountMatrix

MASTviolin(): A ggplot object of MAST violin plots.

MASTregression(): A ggplot object of MAST linear regression plots.

Functions

- `MAST`: Run MAST analysis.
- `thresholdGenes`: Identify adaptive thresholds
- `MASTviolin`: Visualize MAST results using violin plots
- `MASTregression`: Visualize MAST results using linear model plots

Examples

```
data("mouseBrainSubsetSCE")
res <- thresholdGenes(mouseBrainSubsetSCE)
```

---

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

Description

A subset of 30 samples 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

SCtkExperiment

Source

DOI: 10.1126/science.aaa1934

Examples

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

parseRsubreadLogs *Parse Rsubread Logs for Mapping and Feature Count Statistics*

Description

Parse Rsubread Logs for Mapping and Feature Count Statistics

Usage

```
parseRsubreadLogs(alignLog = NULL, featurecountLog = NULL,
  sampleName = NULL)
```

Arguments

alignLog	Path to a log file created by the Rsubread align function
featurecountLog	Path to a log file created by the Rsubread feature count function
sampleName	Sample name corresponding to the two log files

Value

A single line of a data frame with alignment and feature count information

pcaVariances *Get PCA variances*

Description

Get PCA variances

Get PCA variances

Set PCA variances

Usage

```
pcaVariances(x, ...)

## S4 method for signature 'SCtkExperiment'
pcaVariances(x)

## S4 replacement method for signature 'SCtkExperiment'
pcaVariances(x) <- value
```

Arguments

x	SCtkE object
...	other parameters
value	The DataFrame of pcaVariances

Value

A data frame of percent variation explained by each PC.

A SCtkExperiment object with the pcaVariances object set.

Examples

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

`pcaVariances<-` *Set PCA variances*

Description

Set PCA variances

Usage

```
pcaVariances(x, ...) <- value
```

Arguments

`x`	SCtkE object
`...`	other parameters
`value`	PCA variances DataFrame()

Value

A SCtkExperiment object with the pcaVariances slot set.

Examples

```
data("mouseBrainSubsetSCE")
pcaVariances(mouseBrainSubsetSCE)
#getPCA() sets the pcaVariances
newSCE <- getPCA(mouseBrainSubsetSCE, useAssay = "counts")

#alternatively, set the pcaVariances directly
pca <- prcomp(assay(mouseBrainSubsetSCE, "logcounts"))
percentVar <- pca$sdev ^ 2 / sum(pca$sdev ^ 2)
pcaVariances(mouseBrainSubsetSCE) <- DataFrame(percentVar)
```

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 if it is given.

Usage

```
plotBatchVariance(inSCE, useAssay = "logcounts", batch,
  condition = NULL)
```

Arguments

inSCE	Input SCtkExperiment object. Required
useAssay	Indicate which assay to use for PCA. Default is "logcounts"
batch	The column in the annotation data that corresponds to batch. Required
condition	The column in the annotation data that corresponds to condition. Optional

Value

A boxplot of variation explained by batch, condition, and batch+condition (if applicable).

Examples

```
if(requireNamespace("bladderbatch", quietly = TRUE)) {
  library(bladderbatch)
  data(bladderdata)
  dat <- as(as(bladderEset, "SummarizedExperiment"), "SCtkExperiment")
  plotBatchVariance(dat, useAssay="exprs", batch="batch", condition = "cancer")
}
```

plotBiomarker *Given a set of genes, return a ggplot of expression values.*

Description

Given a set of genes, return a ggplot of expression values.

Usage

```
plotBiomarker(inSCE, gene, binary = "Binary", visual = "PCA",
  shape = "No Shape", x = "PC1", y = "PC2", useAssay = "counts",
  reducedDimName = "PCA")
```

Arguments

inSCE	Input SCtkExperiment object. Required
gene	genelist to run the method on.
binary	binary/continuous color for the expression.
visual	Type of visualization (PCA, tSNE or UMAP). Default: "PCA"
shape	shape parameter for the ggplot.
x	x coordinate for PCA
y	y coordinate for PCA
useAssay	Indicate which assay to use. The default is "logcounts".
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.

Value

A Biomarker plot

Examples

```
data("mouseBrainSubsetSCE")
plotBiomarker(mouseBrainSubsetSCE, gene="C1qa", shape="level1class")
```

---

plotDiffEx *Plot Differential Expression*

Description

Plot Differential Expression

Usage

```
plotDiffEx(inSCE, useAssay = "logcounts", condition, geneList,
  clusterRow = TRUE, clusterCol = TRUE, displayRowLabels = TRUE,
  displayColumnLabels = TRUE, displayRowDendrograms = TRUE,
  displayColumnDendrograms = TRUE, annotationColors = NULL,
  scaleExpression = TRUE, columnTitle = "Differential Expression")
```

Arguments

inSCE	Input data object that contains the data to be plotted. Required
useAssay	Indicate which assay to use. Default is "logcounts"
condition	The condition used for plotting the heatmap. Required
geneList	The list of genes to put in the heatmap. Required
clusterRow	Cluster the rows. The default is TRUE
clusterCol	Cluster the columns. The default is TRUE

displayRowLabels
: Display the row labels on the heatmap. The default is TRUE.

displayColumnLabels
: Display the column labels on the heatmap. The default is TRUE

displayRowDendrograms
: Display the row dendrograms on the heatmap. The default is TRUE

displayColumnDendrograms
: Display the column dendrograms on the heatmap. The default is TRUE.

annotationColors
: Set of annotation colors for color bar. If null, no color bar is shown. default is NULL.

scaleExpression
: Row scale the heatmap values. The default is TRUE.

columnTitle
: Title to be displayed at top of heatmap.

Value

ComplexHeatmap object for the provided geneList annotated with the condition.

Examples

```
data("mouseBrainSubsetSCE")
res <- scDiffEx(mouseBrainSubsetSCE,
                useAssay = "logcounts",
                "level1class",
                diffexmethod = "limma")
plotDiffEx(mouseBrainSubsetSCE, condition = "level1class",
           geneList = rownames(res)[1:50], annotationColors = "auto")
```

---

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 SCTKExperiment object. Required.

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")
```

---

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 SCtkExperiment object. Required
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 SCtkExperiment object with saved dimension reduction components or a variable with saved results. 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 SCtkExperiment 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

```
plotUMAP(mouseBrainSubsetSCE, shape = "No Shape", reducedDimName = "UMAP",
         runUMAP = TRUE, useAssay = "counts")
```

`saveBiomarkerRes` *saveBiomarkerRes Save biomarker gene information with a custom name when provided with diffex results.*

Description

saveBiomarkerRes Save biomarker gene information with a custom name when provided with diffex results.

Usage

```
saveBiomarkerRes(inSCE, diffex, biomarkerName, method, ntop = 25,
  logFC = NULL, pVal = NULL)
```

Arguments

inSCE	Input SCtkExperiment object. Required
diffex	results table saved from the differential expression analysis. Required.
biomarkerName	name of the biomarker result to be saved under in rowData(). Required.
method	name of the diffex method used to generate the results. Options are DESeq2, Limma and ANOVA. Required
ntop	number of top N genes. Default is 25. Required
logFC	logfold-change cutoff applied to save biomarker results. Optional
pVal	adjusted p-value cutoff. Optional

Value

a new SCE object with the diffex result saved in the rowData using the "biomarkerName"

Examples

```
data("mouseBrainSubsetSCE")
res <- scDiffEx(mouseBrainSubsetSCE,
                useAssay = "logcounts",
                condition = "level1class",
                ntop = length(rownames(mouseBrainSubsetSCE)),
                usesig = FALSE,
                diffexmethod = "limma")
sceObj <- saveDiffExResults(mouseBrainSubsetSCE, res, "Limma_Level1class", "limma")
bioMarkRes <- saveBiomarkerRes(sceObj, res, "bioMarker", "limma", logFC = 4, pVal = 0.05)
```

---

saveDiffExResults *saveDiffExResults Save Differential Expression Results with a custom name.*

Description

saveDiffExResults Save Differential Expression Results with a custom name.

Usage

```
saveDiffExResults(inSCE, diffex, name, method)
```

Arguments

inSCE	Input SCtkExperiment object. Required
diffex	results table saved from the differential expression analysis. Required
name	name of the result to be saved under in rowData(). Required
method	name of the diffex method used to generate the results. Options are DESeq2, limma and ANOVA. Required

Value

a new SCE object with the diffex result saved in the rowData using the "name"

Examples

```
data("mouseBrainSubsetSCE")
res <- scDiffEx(mouseBrainSubsetSCE,
                useAssay = "logcounts",
                condition = "level1class",
                ntop = length(rownames(mouseBrainSubsetSCE)),
                usesig = FALSE,
                diffexmethod = "limma")
sceObj <- saveDiffExResults(mouseBrainSubsetSCE, res, "Limma_Level1class", "limma")
```

---

scDiffEx *Perform differential expression analysis on a SCtkExperiment object*

Description

Perform differential expression analysis on a SCtkExperiment object

Usage

```
scDiffEx(inSCE, useAssay = "logcounts", condition, covariates = NULL,
  significance = 0.05, ntop = 500, usesig = TRUE, diffexmethod,
  levelofinterest = NULL, analysisType = NULL, controlLevel = NULL,
  adjust = "fdr")

scDiffExDESeq2(inSCE, useAssay = "counts", condition,
  analysisType = "biomarker", levelofinterest = NULL,
  controlLevel = NULL, covariates = NULL, adjust = "fdr")

scDiffExlimma(inSCE, useAssay = "logcounts", condition,
  analysisType = "biomarker", levelofinterest = NULL,
  covariates = NULL, adjust = "fdr")

scDiffExANOVA(inSCE, useAssay = "logcounts", condition,
  covariates = NULL, adjust = "fdr")
```

Arguments

inSCE	Input SCtkExperiment object. Required
useAssay	Indicate which assay to use. Default is "logcounts" for limma and ANOVA, and "counts" for DESeq2.
condition	The name of the condition to use for differential expression. Must be a name of a column from colData that contains at least two labels. Required
covariates	Additional covariates to add to the model. Default is NULL
significance	FDR corrected significance cutoff for differentially expressed genes. Required
ntop	Number of top differentially expressed genes to display in the heatmap. Required
usesig	If TRUE, only display genes that meet the significance cutoff, up to ntop genes. Required

`diffexmethod` The method for performing differential expression analysis. Available options are DESeq2, limma, and ANOVA. Required

`levelofinterest` If the condition has more than two labels, levelofinterest should contain one factor for condition. The differential expression results will use levelofinterest depending on the analysisType parameter.

`analysisType` For conditions with more than two levels, limma and DESeq2 can be run using multiple methods. For DESeq2, choose "biomarker" to compare the levelofinterest to all other samples. Choose "contrast" to compare the levelofinterest to a controlLevel (see below). Choose "fullreduced" to perform DESeq2 in LRT mode. For limma, Choose "biomarker" to compare the levelofinterest to all other samples. Choose "coef" to select a coefficient of interest with levelofinterest (see below). Choose "allcoef" to test if any coefficient is different from zero.

`controlLevel` If the condition has more than two labels, controlLevel should contain one factor from condition to use as the control.

`adjust` Method for p-value correction. See options in p.adjust(). The default is fdr.

Value

A data frame of gene names and adjusted p-values

Functions

- `scDiffExDESeq2`: Perform differential expression analysis with DESeq2
- `scDiffExlimma`: Perform differential expression analysis with limma
- `scDiffExANOVA`: Perform differential expression analysis with ANOVA

Examples

```
data("mouseBrainSubsetSCE")
res <- scDiffEx(mouseBrainSubsetSCE,
                useAssay = "logcounts",
                "level1class",
                diffexmethod = "limma")

data("mouseBrainSubsetSCE")
#sort first 100 expressed genes
ord <- rownames(mouseBrainSubsetSCE)[
  order(rowSums(assay(mouseBrainSubsetSCE, "counts")),
        decreasing = TRUE)][1:100]
#subset to those first 100 genes
subset <- mouseBrainSubsetSCE[ord, ]
res <- scDiffExDESeq2(subset, condition = "level1class")

data("mouseBrainSubsetSCE")
res <- scDiffExlimma(mouseBrainSubsetSCE, condition = "level1class")

data("mouseBrainSubsetSCE")
res <- scDiffExANOVA(mouseBrainSubsetSCE, condition = "level1class")
```

SCtkExperiment *Create a SCtkExperiment*

Description

Create a SCtkExperiment

Usage

```
SCtkExperiment(..., pcaVariances = S4Vectors::DataFrame())
```

Arguments

...	SingleCellExperiment and SummarizedExperiment components
pcaVariances	The percent variation contained in each PCA dimension

Value

A SingleCellExperiment like object with an addition pcaVariances slot.

Examples

```
data("mouseBrainSubsetSCE")
counts_mat <- assay(mouseBrainSubsetSCE, "counts")
sample_annot <- colData(mouseBrainSubsetSCE)
row_annot <- rowData(mouseBrainSubsetSCE)
newSCE <- SCtkExperiment(assays=list(counts=counts_mat),
                         colData=sample_annot,
                         rowData=row_annot)
newSCE <- getPCA(newSCE, useAssay = "counts")
#View the percent variation of the PCA
pcaVariances(newSCE)
```

SCtkExperiment-class *A lightweight S4 extension to the SingleCellExperiment class to store additional information.*

Description

A lightweight S4 extension to the SingleCellExperiment class to store additional information.

Arguments

value	The DataFrame of pcaVariances

Value

A SingleCellExperiment like object with an addition pcaVariances slot.

Slots

pcaVariances The percent variation contained in each PCA dimension

Examples

```
data("mouseBrainSubsetSCE")
counts_mat <- assay(mouseBrainSubsetSCE, "counts")
sample_annot <- colData(mouseBrainSubsetSCE)
row_annot <- rowData(mouseBrainSubsetSCE)
newSCE <- SCtkExperiment(assays=list(counts=counts_mat),
                         colData=sample_annot,
                         rowData=row_annot)
newSCE <- getPCA(newSCE, useAssay = "counts")
#View the percent variation of the PCA
pcaVariances(newSCE)
```

---

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	The input SCtkExperiment class 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

```
#Upload data through the app
if(interactive()){
  singleCellTK()
}

#Load the app with a SCtkExperiment object
if(interactive()){
  data("mouseBrainSubsetSCE")
  singleCellTK(mouseBrainSubsetSCE)
}
```

`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`: Get PCA components for a SCtkE object
- `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)][1:100]
#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)][1:100]
# 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)
```

---

summarizeTable *Summarize SCtkExperiment*

Description

Creates a table of summary metrics from an input SCtkExperiment.

Usage

```
summarizeTable(inSCE, useAssay = "counts", expressionCutoff = 1700)
```

Arguments

inSCE	Input SCtkExperiment object. Required
useAssay	Indicate which assay to summarize. Default is "counts"
expressionCutoff	Count number of samples with fewer than expressionCutoff genes. The default is 1700.

Value

A data.frame object of summary metrics.

Examples

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

---

visPlot *visPlot*

Description

Given a plotting method with condition and gene list, return the respective visualization plot(s).

Usage

```
visPlot(inSCE, useAssay, method, condition, glist, facetWrap = TRUE,
  scaleHMap = TRUE, convertFactor = FALSE)
```

Arguments

inSCE	Input SCtkExperiment object. Required
useAssay	The assay to use in the visualization plot. Required
method	Visualization method. Available options are boxplot, scatterplot, or heatmap. Required
condition	colData annotation of the experiment. Required
glist	selected genes for visualization. Maximum 25 genes. Required
facetWrap	facet wrap according to genes for boxplot, scatterplot and barplot. Default is FALSE. Optional
scaleHMap	scale heatmap expression values. Default is TRUE. Optional
convertFactor	If the condition is not a factor, convert it to a factor before plotting. The default is FALSE

Value

A visualization plot

Examples

```
visPlot(mouseBrainSubsetSCE, "logcounts", "boxplot", "level1class", "C1qa")
visPlot(mouseBrainSubsetSCE, "counts", "scatterplot", "age", "Cmtm5")
visPlot(mouseBrainSubsetSCE, "counts", "heatmap", "level1class",
        c("Cmtm5", "C1qa"))
```

Index