Package 'hammers'

September 15, 2025

Type Package

Title A collection of simple tools addressing common tasks in scRNA-seq analysis

Version 0.1.0

Description hammers is an evolving collection of simple tools designed to address common tasks in scRNA-seq data analysis. The included tools are intended to combine wide applicability with ease of use.

License MIT + file LICENSE

Imports abdiv,
AnnotationDbi,
cluster,
clusterProfiler,
DOSE,
dplyr,
ggplot2,
ggrepel,
grDevices,
henna,
liver,
org.Dr.eg.db,
org.Hs.eg.db,
org.Mm.eg.db,
qs,
rlang,
S4Vectors,
Seurat,
SeuratObject,
SingleCellExperiment,
SummarizedExperiment,
sgof,
stats,
stringr,
text2vec,
withr

Suggests knitr,

rmarkdown,
scater,
scRNAseq,
scuttle,
testthat (>= 3.0.0)

Encoding UTF-8

RoxygenNote 7.3.2

VignetteBuilder knitr

Config/testthat/edition 3

Contents

`addCategory` *Add a categorical column to a data frame based on another column*

Description

This function adds a categorical column to a data frame based on another column.

Usage

```
addCategory(df, col, newCol, hashKeys, hashValues)
```

Arguments

`df`	A data frame.
`col`	Column whose values will be used for creating the new column.
`newCol`	Column to be added.
`hashKeys`	A list of hash keys. If vectors are part of the hash keys, each of their elements will be assigned the corresponding value.
`hashValues`	A vector of hash values. Must have the same length as `hashKeys`.

Value

A data frame with a new categorical column.

Examples

```
df <- data.frame(fruit = c('apple', 'banana', 'cherry', 'grape'))
df <- addCategory(df,
                  'fruit',
                  'color',
                  list(c('apple', 'cherry'),
                  'banana',
                  'grape'),
                  c('red', 'yellow', 'purple'))
```

addMetadataCategory *Add a categorical column to a Seurat metadata or SingleCellExperiment coldata*

Description

Add a categorical column to a Seurat metadata or SingleCellExperiment coldata

Usage

```
addMetadataCategory(
  scObj,
  col,
  newCol,
  hashKeys,
  hashValues,
  newCol2 = NULL,
  hashValues2 = NULL
)
```

Arguments

scObj	A Seurat or SingleCellExperiment object.
col	Column whose values will be used for creating the new column.
newCol	Column to be added.
hashKeys	A list of hash keys. If vectors are part of the hash keys, each of their elements will be assigned the corresponding value.
hashValues	A vector of hash values. Must have the same length as hashKeys.
newCol2	A second column to be added based on the same hash keys. Default is NULL (no second column will be added).
hashValues2	A vector of hash values corresponding to the second column. Default is NULL (no second column will be added).

Value

A Seurat or SingleCellExpression object with one or two new categorical column(s) in the metadata/coldata.

Examples

```
df <- data.frame(fruit = c('apple', 'banana', 'cherry', 'grape'))
df <- addCategory(df,
                  'fruit',
                  'color',
                  list(c('apple', 'cherry'),
                  'banana',
```

```
                'grape'),
                c('red', 'yellow', 'purple'))
```

---

bfCorrectDF *Perform multiple testing correction and filtering with Bonferroni*

Description

This function performs the Bonferroni correction for multiple testing in a dataframe column of p-values and filters the data-frame based on p-values.

Usage

```
bfCorrectDF(df, nTests, pvalThr = 0.05, colStr = "pval", newColStr = "pvalAdj")
```

Arguments

df	A dataframe with a column of p-values.
nTests	Number of tests.
pvalThr	p-value threshold.
colStr	Name of the column of p-values.
newColStr	Name of the column of adjusted p-values that will be created.

Value

The data frame with Benjamini-Yekutieli-corrected p-values.

Examples

```
df <- data.frame(elem = c('A', 'B', 'C', 'D', 'E'),
pval = c(0.032, 0.001, 0.0045, 0.051, 0.048))
bfCorrectDF(df, 5)
```

bhCorrectDF — *Perform multiple testing correction and filtering with Benjamini-Hochberg*

Description

This function performs the Benjamini-Hochberg correction for multiple testing in a dataframe column of p-values and filters the data-frame based on p-values.

Usage

```
bhCorrectDF(df, ...)
```

Arguments

df	A dataframe with a column of p-values.
...	Additional arguments passed to fdrCorrectDF.

Value

The data frame with Benjamini-Hochberg-corrected p-values.

Examples

```
df <- data.frame(elem = c('A', 'B', 'C', 'D', 'E'),
pval = c(0.032, 0.001, 0.0045, 0.051, 0.048))
bhCorrectDF(df)
```

byCorrectDF — *Perform multiple testing correction and filtering with Benjamini-Yekutieli*

Description

This function performs the Benjamini-Yekutieli correction for multiple testing in a dataframe column of p-values and filters the data-frame based on p-values.

Usage

```
byCorrectDF(df, ...)
```

Arguments

df	A dataframe with a column of p-values.
...	Additional arguments passed to fdrCorrectDF.

Value

The data frame with Benjamini-Yekutieli-corrected p-values.

Examples

```
df <- data.frame(elem = c('A', 'B', 'C', 'D', 'E'),
pval = c(0.032, 0.001, 0.0045, 0.051, 0.048))
byCorrectDF(df)
```

---

centerOfMass *Calculate the coordinates of the center of mass*

Description

This function calculates the coordinates of the center of mass based on a matrix of cell embeddings and a vector of weights.

Usage

```
centerOfMass(dimMat, weights)
```

Arguments

dimMat	A matrix of cell embeddings.
weights	A vector of weights.

Value

A vector containing the coordinates of the center of mass.

Examples

```
dimMat <- matrix(data=c(2, 3, 1, 3, 6, 8), nrow=3, ncol=2)
weights <- c(0.8, 6, 16)
centerOfMass(dimMat, weights)
```

checkGenes *Check if all genes exist in the single-cell expression object*

Description

This function checks if all genes exist in the single-cell expression object.

Usage

```
checkGenes(scObj, genes)
```

Arguments

scObj	A Seurat, SingleCellExperiment, dgCMatrix or matrix object.
genes	A character vector of genes.

Value

None. This function is called for its side effect.

Examples

```
scObj <- scRNAseq::BaronPancreasData('human')
checkGenes(scObj, c('AURKA', 'TOP2A', 'MKI67'))
```

colCenters *Calculate the coordinates of centers of mass of metadata/coldata columns*

Description

This function calculates the coordinates of the center of mass of the selected metadata/coldata columns from a Seurat or SingleCellExpression object.

Usage

```
colCenters(scObj, columns)
```

Arguments

scObj	A Seurat or SingleCellExperiment object.
columns	Numeric columns.

Value

A data frame containing the coordinates of centers of mass.

colsDimPlot *Plot Seurat DimPlot with added labeled points for numeric columns*

Description

This function plots a Seurat DimPlot with added labeled points for metadata numeric columns.

Usage

```
colsDimPlot(seuratObj, cols, ...)
```

Arguments

seuratObj	A Seurat object.
cols	Genes whose centers of mass will be plotted.
...	Additional parameters passed to pointsDimPlot.

Value

A ggplot object.

Examples

```
sceObj <- scRNAseq::BaronPancreasData('human')
sceObj <- scuttle::logNormCounts(sceObj)
seuratObj <- suppressWarnings(Seurat::as.Seurat(sceObj))
seuratObj <- Seurat::FindVariableFeatures(seuratObj)
seuratObj <- Seurat::ScaleData(seuratObj)
seuratObj <- Seurat::RunPCA(seuratObj)
seuratObj <- suppressWarnings(Seurat::RunUMAP(seuratObj, dims=1:15))
colsDimPlot(seuratObj, c('nCount_originalexp', 'nFeature_originalexp'))
```

computeSilhouette *Compute cluster silhouette for single-cell expression object*

Description

This function computes the silhouette for each cell in the Seurat or SingleCellExperiment object.

Usage

```
computeSilhouette(scObj, idClass, pcaMat = NULL)
```

Arguments

scObj	A Seurat or SingleCellExperiment object.
idClass	Identity class. Must be present among the metadata columns of the single-cell expression object.
pcaMat	PCA matrix.

Value

The input object (Seurat or SingleCellExperiment) with an added metadata silhouette column.

Examples

```
scObj <- scRNAseq::BaronPancreasData('human')
scObj <- scuttle::logNormCounts(scObj)
scObj <- scater::runPCA(scObj)
scObj <- computeSilhouette(scObj, 'label')
head(scCol(scObj, 'silhouette'))
```

---

devPlot.default *Saves plot or list of plots*

Description

This function saves a plot or list of plots as a pdf. Can also take as input a function that returns a ggplot object together with its arguments.

Usage

```
## Default S3 method:
devPlot(plotObject, ...)

## S3 method for class '`function`'
devPlot(plotObject, ...)

## S3 method for class 'ggplot'
devPlot(plotObject, ...)

## S3 method for class 'list'
devPlot(plotObject, ...)

devPlot(plotObject, ...)
```

Arguments

plotObject	A function, ggplot object, or list of ggplot objects.
...	Additional arguments.

Value

No value. This function is called for its side effect.

Examples

```
library(ggplot2)
df <- data.frame(x = c(1, 2), y = c(3, 5))
p <- ggplot(df) + geom_point(aes(x, y))
devPlot(p)

simplePlot <- function(df, title)
    return(ggplot(df) + geom_point(aes(x, y)) + ggtitle(title))

devPlot(simplePlot, df, 'Plot title')
```

---

distributionPlot *Plot the distribution of cells across two columns*

Description

This function plots the distribution of cells across two columns.

Usage

```
distributionPlot(
  scObj,
  plotTitle = "Distribution plot",
  col1 = "seurat_clusters",
  col2 = "orig.ident",
  xLab = "Column 1",
  yLab = "Count",
  legendLab = "Column 2",
  palette = "Spectral"
)
```

Arguments

scObj	A Seurat or SingleCellExperiment object.
plotTitle	Plot title.
col1	Column as string.
col2	Column as string.
xLab	x axis label.
yLab	y axis label.
legendLab	Legend label.
palette	Color palette.

Value

A ggplot object.

Examples

```
scObj <- scRNAseq::BaronPancreasData('human')
distributionPlot(scObj, col1='donor', col2='label')
```

`geneCenters` *Calculate the coordinates of centers of mass of gene expression*

Description

This function calculates the coordinates of the center of mass of the expression of input genes.

Usage

```
geneCenters(scObj, genes)
```

Arguments

`scObj`	A `Seurat`, `SingleCellExperiment`, `dgCMatrix` or `matrix` object.
`genes`	Selected genes. If `NULL`, all genes will be retained

Value

A data frame containing the coordinates of centers of mass.

Examples

```
sceObj <- scRNAseq::BaronPancreasData('human')
sceObj <- scuttle::logNormCounts(sceObj)
sceObj <- scater::runUMAP(sceObj)
geneCenters(sceObj, c('AURKA', 'MKI67', 'TOP2A'))
```

genesDimPlot *Plot Seurat DimPlot with added labeled points for genes*

Description

This function plots a Seurat DimPlot with added labeled points for genes.

Usage

```
genesDimPlot(seuratObj, genes, ...)
```

Arguments

seuratObj	A Seurat object.
genes	Genes whose centers of mass will be plotted.
...	Additional parameters passed to pointsDimPlot.

Value

A ggplot object.

Examples

```
sceObj <- scRNAseq::BaronPancreasData('human')
sceObj <- scuttle::logNormCounts(sceObj)
seuratObj <- suppressWarnings(Seurat::as.Seurat(sceObj))
seuratObj <- Seurat::FindVariableFeatures(seuratObj)
seuratObj <- Seurat::ScaleData(seuratObj)
seuratObj <- Seurat::RunPCA(seuratObj)
seuratObj <- suppressWarnings(Seurat::RunUMAP(seuratObj, dims=1:15))
genesDimPlot(seuratObj, c('AURKA', 'TOP2A', 'MKI67'))
```

genesER *Perform enrichment analysis on a set of genes*

Description

This function performs enrichment analysis on a set of genes.

Usage

```
genesER(genes, species, funString = c("enrichGO", "enrichKEGG", "enrichWP"))
```

Arguments

genes	A character vector of gene symbols.
species	Species. Must be one of 'human', 'mouse' and 'zebrafish'.
funString	Name of enrichment function from clusterProfiler. Must be a character selected from 'enrichGO', 'enrichKEGG' and 'enrichWP'.

Value

Enrichment result.

Examples

```
m <- genesER(c('AURKA', 'TOP2A', 'CENPF', 'PTTG2', 'MKI67', 'BIRC5', 'RRM2'),
'human')
```

---

hashMap *Create a hash map*

Description

This function creates a hash map.

Usage

```
hashMap(hashKeys, hashValues)
```

Arguments

hashKeys	A list of hash keys. If vectors are part of the hash keys, each of their elements will be assigned the corresponding value.
hashValues	A vector of hash values. Must have the same length as hashKeys.

Value

A named vector.

Examples

```
hashMap(list(2, c(3, 4, 5), 6, 8), c('a', 'b', 'c', 'd'))
```

joinCharCombs *Join all combinations of elements from character vectors*

Description

This function joins all combinations of elements from character vectors with a separating character.

Usage

```
joinCharCombs(..., joinChar = "_")
```

Arguments

...	Vectors passed to expandGrid.
joinChar	Character used to join combinations.

Value

A character vector.

Examples

```
joinCharCombs(c('a', 'b', 'c', 'd'), c('eee', 'ff'), c(1, 2, 3))
```

metadataDF.default *Extract metadata from object as a data frame*

Description

This function extracts the metadata from a Seurat or SingleCellExperiment object as a data frame.

Usage

```
## Default S3 method:
metadataDF(scObj)

## Default S3 replacement method:
metadataDF(scObj) <- value

## S3 method for class 'Seurat'
metadataDF(scObj)

## S3 replacement method for class 'Seurat'
metadataDF(scObj) <- value
```

```
## S3 method for class 'SingleCellExperiment'
metadataDF(scObj)

## S3 replacement method for class 'SingleCellExperiment'
metadataDF(scObj) <- value

metadataDF(scObj)

metadataDF(scObj) <- value
```

Arguments

scObj	A Seurat or SingleCellExperiment object.
value	A data frame to replace metadata with.

Value

A metadata data frame.

Examples

```
scObj <- scRNAseq::BaronPancreasData('human')
df <- metadataDF(scObj)
```

---

metadataNames.default *Return metadata names.*

Description

This function extracts metadata names from a Seurat or SingleCellExperiment object.

Usage

```
## Default S3 method:
metadataNames(scObj)

## S3 method for class 'Seurat'
metadataNames(scObj)

## S3 method for class 'SingleCellExperiment'
metadataNames(scObj)

metadataNames(scObj)
```

Arguments

scObj	A Seurat or SingleCellExperiment object.

Value

The names of the metadata columns.

Examples

```
scObj <- scRNAseq::BaronPancreasData('human')
colNames <- metadataNames(scObj)
```

`nearestNeighbors` *Get nearest neighbors from distance matrix*

Description

This function gets the nearest neighbors from a distance matrix.

Usage

```
nearestNeighbors(distMat)
```

Arguments

`distMat`	A distance matrix.

Value

A named character vector.

Examples

```
df <- data.frame(v = c(1, 2, 4, 5, 6),
w = c(2, 3, 1, 5, 8),
x = c(2, 8, 7, 1, 1),
y = c(2, 3, 2, 2, 4),
z = c(1, 9, 9, 7, 6))
distMat <- as.matrix(stats::dist(df))
rownames(distMat) <- c('v', 'w', 'x', 'y', 'z')
colnames(distMat) <- c('v', 'w', 'x', 'y', 'z')
nearestNeighbors(distMat)
```

normalizeSilhouette *Normalize silhouette by identity class for single-cell expression object*

Description

This function normalizes the already computed silhouette for each identity class in the single-cell expression object.

Usage

```
normalizeSilhouette(scObj, idClass)
```

Arguments

scObj	A Seurat or SingleCellExperiment object.
idClass	Identity class. Must be present among the metadata columns of the single-cell expression object.

Value

A data frame with normalized silhouettes for each unique element in the identity class.

pointsDimPlot *Plot Seurat DimPlot with added labeled points*

Description

This function plots a Seurat DimPlot with added labeled points.

Usage

```
pointsDimPlot(
  seuratObj,
  plotTitle = "Dim plot",
  pointsDF = NULL,
  pointShape = 4,
  pointSize = 2,
  labelSize = 2.5,
  maxOverlaps = 30,
  ...
)
```

Arguments

seuratObj	A Seurat object.
plotTitle	Plot title.
pointsDF	A data frame of points with two columns representing the x and y coordinates.
pointShape	Point shape.
pointSize	Point size.
labelSize	Label size.
maxOverlaps	Maximum overlaps.
...	Additional parameters passed to Seurat::DimPlot.

Value

A ggplot object.

Examples

```
sceObj <- scRNAseq::BaronPancreasData('human')
sceObj <- scuttle::logNormCounts(sceObj)
seuratObj <- suppressWarnings(Seurat::as.Seurat(sceObj))
seuratObj <- Seurat::FindVariableFeatures(seuratObj)
seuratObj <- Seurat::ScaleData(seuratObj)
seuratObj <- Seurat::RunPCA(seuratObj)
seuratObj <- suppressWarnings(Seurat::RunUMAP(seuratObj, dims=1:15))
pointsDF <- data.frame(x = c(2, 3),
y = c(1, 6),
row.names = c('P1', 'P2'))
pointsDimPlot(seuratObj, pointsDF=pointsDF)
```

---

proximity *Compute proximity between two vectors based on Euclidean distance*

Description

This functions computes proximity between two vectors based on Euclidean distance and an input maximum distance.

Usage

```
proximity(x, y, maxDist)
```

Arguments

x	A numeric vector.
y	A numeric vector.
maxDist	Maximum distance.

Value

A number between 0 and 1.

Examples

```
proximity(2, 3, 6)
```

`pvalRiverPlot` *Plot representation data frame*

Description

This function plots representation data frame as an alluvial plot.

Usage

```
pvalRiverPlot(df, weightExp = 1/2, ...)
```

Arguments

`df`	A data frame.
`weightExp`	Exponent used in constructing weight from p-values.
`...`	Additional parameters passed to `henna::riverPlot`

Value

A ggplot object

Examples

```
scObj <- scRNAseq::BaronPancreasData('human')
df <- repAnalysis(scObj, 'donor', 'label')
pvalRiverPlot(df)
```

qGrab *Read and delete a .qs file*

Description

This functions reads a .qs file, deletes it, and returns its content.

Usage

```
qGrab(qsFile)
```

Arguments

qsFile	Name of .qs file with path.

Value

The content of the .qs file.

Examples

```
library(qs)
qsave(c(1, 2, 3), 'temp.qs')
qGrab('temp.qs')
```

repAnalysis *Find the differential representation of two Seurat columns*

Description

This function find the differential representation of two Seurat columns.

Usage

```
repAnalysis(
  scObj,
  col1 = "seurat_clusters",
  col2 = "orig.ident",
  doOverrep = TRUE,
  fdrMethod = c("BY", "BH"),
  pvalThr = 0.05
)
```

Arguments

scObj	A Seurat or SingleCellExperiment object.
col1	Column as string.
col2	Column as string.
doOverrep	Whether to perform overrepresentation analysis. If FALSE, underrepresentation analysis will be performed instead.
fdrMethod	False discovery rate control method. Options are 'BY' (Benjamini-Yekutieli) and 'BH' (Benjamini-Hochberg).
pvalThr	p-value threshold.

Value

An overrepresentation or underrepresentation data frame.

Examples

```
sceObj <- scRNAseq::BaronPancreasData('human')
repAnalysis(sceObj, 'donor', 'label')
```

---

safeMessage *Message an input if verbose is set to TRUE*

Description

This function messages an input if verbose is set to TRUE.

Usage

```
safeMessage(msg, verbose = TRUE)
```

Arguments

msg	Message
verbose	Whether the message should be displayed.

Value

No return value. This function is called for its side effect (messaging the input if verbose is set to TRUE).

Examples

```
safeMessage('message')
```

safeMinmax *Perform min-max normalization when possible; otherwise return a single-value vector.*

Description

This function min-max-normalizes a vector when possible, and otherwise returns a single-value vector.

Usage

```
safeMinmax(scores, safeVal = 0)
```

Arguments

scores	Numeric vector.
safeVal	Value to replace all values with when all values in the vector are the same.

Value

Min-max-normalized scores or a single-value vector.

Examples

```
safeMinmax(c(0, 3, 2, 1, 4, 5.5, 6.32, 8, 1.1))
```

scCol.default *Extract a metadata/coldata column from object.*

Description

This function extracts a metadata/coldata column from a Seurat or SingleCellExperiment object.

Usage

```
## Default S3 method:
scCol(scObj, col)

## S3 method for class 'Seurat'
scCol(scObj, col)

## S3 method for class 'SingleCellExperiment'
scCol(scObj, col)

scCol(scObj, col)
```

Arguments

scObj	A Seurat or SingleCellExperiment object.
col	Column name.

Value

A vector.

Examples

```
scObj <- scRNAseq::BaronPancreasData('human')
v <- scCol(scObj, 'label')
```

---

scColCounts *Extract count information from single-cell expression object column*

Description

This function extracts count information from the column of a Seurat or SingleCellExperiment object.

Usage

```
scColCounts(scObj, col = "orig.ident")
```

Arguments

scObj	A Seurat or SingleCellExperiment object.
col	Column as string.

Value

A frequency vector with the unique column values as names.

Examples

```
scObj <- scRNAseq::BaronPancreasData('human')
scColCounts(scObj, 'label')
```

scColPairCounts *Extract count information from Seurat column*

Description

This function extracts count information from Seurat column.

Usage

```
scColPairCounts(scObj, col1 = "seurat_clusters", col2 = "orig.ident")
```

Arguments

scObj	A Seurat or SingleCellExperiment object.
col1	Column as string.
col2	Column as string.

Value

A data frame listing the counts of all combinations of pairs from two categorical columns.

Examples

```
scObj <- scRNAseq::BaronPancreasData('human')
scColPairCounts(scObj, 'donor', 'label')
```

scExpMat.default *Extracts the expression matrix from object.*

Description

This function extracts an expression matrix from a Seurat or SingleCellExperiment object.

Usage

```
## Default S3 method:
scExpMat(
  scObj,
  dataType = c("data", "counts", "logcounts"),
  genes = NULL,
  densify = TRUE
)

## S3 method for class 'Seurat'
```

```
scExpMat(
  scObj,
  dataType = c("data", "counts", "logcounts"),
  genes = NULL,
  densify = TRUE
)

## S3 method for class 'SingleCellExperiment'
scExpMat(
  scObj,
  dataType = c("data", "counts", "logcounts"),
  genes = NULL,
  densify = TRUE
)

## S3 method for class 'dgCMatrix'
scExpMat(
  scObj,
  dataType = c("data", "counts", "logcounts"),
  genes = NULL,
  densify = TRUE
)

## S3 method for class 'matrix'
scExpMat(
  scObj,
  dataType = c("data", "counts", "logcounts"),
  genes = NULL,
  densify = TRUE
)

scExpMat(
  scObj,
  dataType = c("data", "counts", "logcounts"),
  genes = NULL,
  densify = TRUE
)
```

Arguments

scObj	A Seurat, SingleCellExperiment, dgCMatrix or matrix object.
dataType	Expression data type. Ignored if scObj is of class dgCMatrix or matrix.
genes	Selected genes. If NULL, all genes will be retained
densify	Whether to convert to dense matrix.

Value

An expression matrix.

Examples

```
scObj <- scRNAseq::BaronPancreasData('human')
mat <- scExpMat(scObj, 'counts')
```

`scGeneExp.default` *Extracts the expression of a single gene*

Description

This function extracts the expression of a single gene from a Seurat, SingleCellExperiment, dgCMatrix or matrix object.

Usage

```
## Default S3 method:
scGeneExp(scObj, gene, dataType = c("counts", "data", "logcounts"))

## S3 method for class 'Seurat'
scGeneExp(scObj, gene, dataType = c("counts", "data", "logcounts"))

## S3 method for class 'SingleCellExperiment'
scGeneExp(scObj, gene, dataType = c("counts", "data", "logcounts"))

## S3 method for class 'dgCMatrix'
scGeneExp(scObj, gene, dataType = c("counts", "data", "logcounts"))

## S3 method for class 'matrix'
scGeneExp(scObj, gene, dataType = c("counts", "data", "logcounts"))

scGeneExp(scObj, gene, dataType = c("data", "counts", "logcounts"))
```

Arguments

`scObj`	A `Seurat`, `SingleCellExperiment`, `dgCMatrix` or `matrix` object.
`gene`	Selected gene.
`dataType`	Expression data type. Ignored if `scObj` is of class `dgCMatrix` or `matrix`.

Value

A gene expression vector.

Examples

```
scObj <- scRNAseq::BaronPancreasData('human')
v <- scGeneExp(scObj, 'AURKA')
```

scPCAMat *Extracts the PCA matrix from object.*

Description

This function extracts the PCA matrix from a Seurat or SingleCellExperiment object.

Usage

```
scPCAMat(scObj)
```

Arguments

scObj	A Seurat or SingleCellExperiment object.

Value

A PCA matrix.

Examples

```
scObj <- scRNAseq::BaronPancreasData('human')
scObj <- scuttle::logNormCounts(scObj)
scObj <- scater::runPCA(scObj)
pcaMat <- scPCAMat(scObj)
```

scUMAPMat *Extracts the UMAP matrix from object.*

Description

This function extracts the UMAP matrix from a Seurat or SingleCellExperiment object.

Usage

```
scUMAPMat(scObj)
```

Arguments

scObj	A Seurat or SingleCellExperiment object.

Value

A UMAP matrix.

Examples

```
scObj <- scRNAseq::BaronPancreasData('human')
scObj <- scuttle::logNormCounts(scObj)
scObj <- scater::runUMAP(scObj)
umapMat <- scUMAPMat(scObj)
```

`shuffleGenes` *Replaces genes from vector*

Description

This function removes and adds genes from vector at random.

Usage

```
shuffleGenes(
  scObj,
  genes,
  lossFrac,
  noiseFrac,
  geneCountThresh = 10,
  seed = 1,
  verbose = TRUE
)
```

Arguments

`scObj`	A `Seurat` or `SingleCellExperiment` object.
`genes`	A character vector.
`lossFrac`	Fraction of genes than be removed. Must be in `[0, 1]`.
`noiseFrac`	Amount of noise (random genes) in the final gene vector. Must be in `[0, 1)`
`geneCountThresh`	Minimum number of cells in which newly added genes must be expressed.
`seed`	Random seed.
`verbose`	Whether the output should be verbose.

Value

Genes vector after changes.

Examples

```
scObj <- scRNAseq::BaronPancreasData('human')
genes <- c('TOP2A', 'BIRC5', 'MKI67', 'RRM2', 'CENPF', 'PTTG2', 'CLSPN')
shuffleGenes(scObj, genes, 0.3, 0.9)
```

`tabulateVector` *Convert a vector to a data frame based on input row and column names*

Description

This function converts a vector to a data frame based on input row and column names. Optionally, it also calculates the row means.

Usage

```
tabulateVector(
  v,
  rowNames,
  colNames,
  addRowMeans = FALSE,
  sortByRowMeans = FALSE
)
```

Arguments

`v`	A vector.
`rowNames`	A character vector.
`colNames`	A character vector.
`addRowMeans`	Whether to add the row means to the data frame.
`sortByRowMeans`	Whether to sort by row means.

Value

A data frame.

Examples

```
v <- c(2, 3, 4, 19, 15, 25, 32, 8)
res <- tabulateVector(v, paste0('r', seq(4)), paste0('c', seq(2)))
```

termGenes *Extract genes enriched for terms*

Description

This function extracts genes enriched for terms from an `enrichResult` object.

Usage

```
termGenes(er, terms, negTerms = NULL)
```

Arguments

`er`	Enrichment result.
`terms`	Terms for which enriched genes should be extracted.
`negTerms`	Terms for which enriched genes should be subtracted from the genes enriched for `terms`.

Value

Genes enriched for terms.

Examples

```
m <- genesER(c('AURKA', 'TOP2A', 'CENPF', 'PTTG2', 'MKI67', 'BIRC5', 'RRM2'),
'human')
termGenes(m, 'chromosome segregation', 'meiosis I')
```

timeFun *Time a function*

Description

This function prints the time required to run a function and returns the function output.

Usage

```
timeFun(fun, ...)
```

Arguments

`fun`	A function.
`...`	Additional parameters passed to the function.

Value

The function output.

Examples

```
res <- timeFun(sum, 2, 3, 4)
```

Index