Package: broadSeq (via r-universe)

June 21, 2026

Title broadSeq : for streamlined exploration of RNA-seq data

Version 1.6.0

Description This package helps user to do easily RNA-seq data analysis with multiple methods (usually which needs many different input formats). Here the user will provid the expression data as a SummarizedExperiment object and will get results from different methods. It will help user to quickly evaluate different methods.

License MIT + file LICENSE

URL `https://github.com/dasroy/broadSeq`

BugReports `https://github.com/dasroy/broadSeq/issues`

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

biocViews GeneExpression, DifferentialExpression, RNASeq, Transcriptomics, Sequencing, Coverage, GeneSetEnrichment, GO

Suggests knitr, limma (>= 3.54.0), rmarkdown, stats (>= 4.2.2), samr

Depends dplyr, ggpubr, SummarizedExperiment

Imports BiocStyle, DELocal, EBSeq (>= 1.38.0), DESeq2 (>= 1.38.2), NOISeq, forcats (>= 1.0.0), genefilter, ggplot2, ggplotify, plyr, clusterProfiler (>= 4.8.2), pheatmap, sechm (>= 1.6.0), stringr, purrr (>= 0.3.5), edgeR (>= 3.40.1)

VignetteBuilder knitr

Config/pak/sysreqs libcairo2-dev cmake libfontconfig1-dev libfreetype6-dev libglpk-dev make libicu-dev libpng-dev libuv1-dev libxml2-dev libssl-dev perl libnode-dev zlib1g-dev

Repository https://bioc-release.r-universe.dev

Date/Publication 2026-04-28 13:03:13 UTC

RemoteUrl https://github.com/bioc/broadSeq

RemoteRef RELEASE_3_23

RemoteSha 332bb5ca54fb6b44188760cdcf57ec12a79e9f61

Contents

`broadSeq-package` *broadSeq : for streamlined exploration of RNA-seq data*

Description

This package helps user to do easily RNA-seq data analysis with multiple methods (usually which needs many different input formats). Here the user will provid the expression data as a SummarizedExperiment object and will get results from different methods. It will help user to quickly evaluate different methods.

Author(s)

Maintainer: Rishi Das Roy <`rishi.dasroy@gmail.com`> (ORCID)

See Also

Useful links:

- `https://github.com/dasroy/broadSeq`
- Report bugs at `https://github.com/dasroy/broadSeq/issues`

`combinedEnrichment` *Provides GO gene set enrichment and over-representation analysis*

Description

This wrapper function combines clusterProfiler::gseGO and clusterProfiler::enrichGO. The input type of thes two methods are different; order ranked geneList and a vector of entrez gene id. Here combinedEnrichment function internally generates these two data types from user defined DEG_table (differentially expresssed genes).

Usage

```
combinedEnrichment(
  DEG_table,
  geneCol = "ID",
  logCol = "logFoldChange",
  OrgDB = "org.Hs.eg.db",
  keyType,
  universe,
  ont = "BP",
  logfoldCut = 1,
  pvalueCutoff = 0.05,
  qvalueCutoff = 0.05
)
```

Arguments

`DEG_table`	A data.frame atleast with two columns.
`geneCol`	The column name of DEG_table which provides gene ids and should be compatible with keytype parameter.
`logCol`	The column name of DEG_table which provides logfold(numeric) values to create a order ranked geneList for gseGO funtion.
`OrgDB`	OrgDb; passed to clusterProfiler functions
`keyType`	keytype of input gene(geneCol). One of the keytypes(OrgDB); passed to clusterProfiler functions
`universe`	background genes; passed to clusterProfiler::enrichGO.
`ont`	one of "BP", "MF", and "CC" subontologies, or "ALL" for all three.; passed to clusterProfiler functions
`logfoldCut`	to filter genes based on parameter logCol
`pvalueCutoff`	; passed to clusterProfiler functions
`qvalueCutoff`	; passed to clusterProfiler functions

Value

a named list of three data.frames which are output of gseGO("gseResult") and enrichGO ("oraUP" and "oraDOWN").

`genes_plot` *Expression of multiple genes/features from a single assay as boxplot (or added dotplot)*

Description

Expression of multiple genes/features from a single assay as boxplot (or added dotplot)

Boxplot of a single gene/feature from multiple assays

Usage

```
genes_plot(se, features, assayName = "counts", facet.by = "feature", x, ...)

assay_plot(se, feature, assayNames = c("counts"), x, ...)
```

Arguments

`se`	Object of `SummarizedExperiment` class
`features`	a character vector of rownames or named list of character vectors where name is one of the colnames of rowdata.
`assayName`	One of the values from SummarizedExperiment::assayNames(se); default is "counts" assay
`facet.by`	must be one of the column names of rowData(se). default "feature" which is equivalent to rownames of rowData
`x`	a column name of colData which will be used in x-axis
`...`	other arguments to be passed to ggpubr::`ggboxplot`
`feature`	a character vector of rownames or named list of character vectors where name is one of the column of rowdata.
`assayNames`	names from SummarizedExperiment::assayNames(se); default value is "counts"

Value

ggplot object

return an object of class ggarrange, which is a ggplot or a list of ggplot.

Examples

```
se <- readRDS(system.file("extdata", "rat_vole_mouseSE_salmon.rds",
    package = "broadSeq"))
# The normalized values are added with the assay name "logCPM"
se <- broadSeq::normalizeEdgerCPM(se ,method = "none",cpm.log = TRUE )

broadSeq::genes_plot(se,
                     features = list(mouse_gene_id = c("ENSMUSG00000022510" ,
                                                       "ENSMUSG00000027985")),
```

```
                     facet.by = "symbol", # column of rowData
                     x = "stage",  fill="stage")

broadSeq::genes_plot(se,
                     features = list(symbol=c("Shh","Edar") ),
                     facet.by = "symbol", # column of rowData
                     x = "stage",  fill="stage")

broadSeq::assay_plot(se, feature = c("Shh"),
                    assays =  c("counts","logCPM"),
                    x = "stage", fill="stage", add="dotplot", palette = "npg")
```

`normalizeEdgerCPM` *Use of edgeR package to normalize count data*

Description

Use of edgeR package to normalize count data

Usage

```
normalizeEdgerCPM(se, method = "TMM", cpm.log = TRUE, ...)
```

Arguments

`se`	Object of `SummarizedExperiment` class
`method`	value for edgeR::`normLibSizes` function. default "TMM"
`cpm.log`	value for edgeR::`cpm` function. default TRUE
`...`	passed to normLibSizes function

Value

Object of `SummarizedExperiment` class where a new assay is added to the input object.

Examples

```
se <- readRDS(system.file("extdata",
        "rat_vole_mouseSE_salmon.rds",
        package = "broadSeq"))

se <- broadSeq::normalizeEdgerCPM(se , method = "TMM", cpm.log = FALSE )
# The normalized values are added with the assay name "TMM"
SummarizedExperiment::assayNames(se)
```

`plot_MDS` *Classical multidimensional scaling*

Description

Classical multidimensional scaling is based on measuring the distance between the samples.

Usage

```
plot_MDS(se, scaledAssay = "vst", ntop = 500L, features = NULL, ...)
```

Arguments

`se`	Object of `SummarizedExperiment` class
`scaledAssay`	an scaled assay name from SummarizedExperiment::assayNames(se)
`ntop`	number of most-variable genes to select. Igored if "features" is specified.
`features`	character vector features/genes to be used to measure distance between the samples
`...`	other arguments like color or shape whose values should be similar to colData columns names passed to ggpubr::`ggscatter`

Value

ggplot object

Examples

```
se <- readRDS(system.file("extdata","rat_vole_mouseSE_salmon.rds", package = "broadSeq"))

se <- broadSeq::transformDESeq2(se,method = "vst"  )

broadSeq::plot_MDS(se, scaledAssay = "vst", ntop=500,
                    color = "species", shape = "stage",
                    ellipse=TRUE, legend = "bottom")
```

`plotHeatmapCluster` *Plot clustered heatmaps*

Description

Plot clustered heatmaps from SummarizedExperiment with pheatmap and return object as ggplot

Usage

```
plotHeatmapCluster(
  se,
  scaledAssay = "vst",
  ntop = 500L,
  features = NULL,
  show_geneAs = NULL,
  annotation_col = NA,
  annotation_row = NA,
  ...
)
```

Arguments

`se`	Object of `SummarizedExperiment` class
`scaledAssay`	an scaled assay name from SummarizedExperiment::assayNames(se)
`ntop`	number of most-variable genes to select. Igored if "features" is specified.
`features`	character vector features/genes to be used to measure distance between the samples
`show_geneAs`	a character vector of colnames of rowData(se)
`annotation_col`	a character vector of colnames of colData(se)
`annotation_row`	a list of character vector of colnames of rowData(se)
`...`	other arguments like color or shape whose values should be similar to colData columns names passed to `pheatmap`

Value

ggplot object

Examples

```
se <- readRDS(system.file("extdata","rat_vole_mouseSE_salmon.rds", package = "broadSeq"))

se <- broadSeq::normalizeEdgerCPM(se ,method = "none",cpm.log = TRUE )

broadSeq::plotHeatmapCluster(
    se,
    scaledAssay = "logCPM",
    annotation_col = c("species", "stage"),
    annotation_row = c("Class","gene_biotype"),
    ntop = 30, show_geneAs = "symbol",
    cluster_cols = TRUE, cluster_rows = FALSE,
    show_rownames = TRUE, show_colnames = FALSE,
    main = "Top 30 variable gene vst"
)
```

`prcompTidy` *Perform Principal Components Analysis*

Description

This function returns the results of stats::`prcomp` in a tidy list format. This is more flexible for further custom PCA , biplot and exploring gene(factor) loading of the PCA.

Usage

```
prcompTidy(se, scaledAssay = "vst", ntop = 500L, features = NULL, ...)

plotAnyPC(computedPCA, x = 1, y = 2, ...)

biplotAnyPC(computedPCA, x = 1, y = 2, genes = NULL, genesLabel = NULL, ...)

getFeatureLoadRanking(computedPCA, pcs = seq_len(5), topN = 10, keep)
```

Arguments

`se`	Object of `SummarizedExperiment` class
`scaledAssay`	an scaled assay name from SummarizedExperiment::assayNames(se)
`ntop`	number of most-variable genes to select. Igored if "features" is specified.
`features`	character vector features/genes to be used for PCA
`...`	other arguments like color or shape whose values should be similar to colData columns names passed to ggpubr::`ggscatter`
`computedPCA`	a list of data.frame returned by `prcompTidy`
`x`	PC number for x-axis default 1
`y`	PC number for y-axis default 2
`genes`	if genes is NULL then top max and min loaded genes of each PCs are plotted
`genesLabel`	one of rowData column names
`pcs`	The numbers of PCs
`topN`	Number of features per PC
`keep`	the column names of rowData to keep the corresponding information

Details

Reused code

Value

a list with four `data.frame` objects: pc_scores, eigen_values, loadings (eigen vectors) and the original data.

ggplot object

ggplot object

a data.frame

Examples

```
se <- readRDS(system.file("extdata","rat_vole_mouseSE_salmon.rds", package = "broadSeq"))

se <- broadSeq::normalizeEdgerCPM(se ,method = "none",cpm.log = TRUE )
computedPCA_logCPM <- broadSeq::prcompTidy(se, scaledAssay = "logCPM", ntop = 500)

plotAnyPC(computedPCA = computedPCA_logCPM, x = 1, y = 2, color = "species",
         shape = "stage",legend = "bottom")
plotAnyPC(computedPCA = computedPCA_logCPM, x = 2, y = 3, color = "species",
         shape = "stage",legend = "bottom")

computedPCA_logCPM$eigen_values %>%
 dplyr::filter(var_exp >= 0.5) %>% # Selecting PC explaining more than 1% variance
    ggbarplot(x="PC",y="var_exp", label = TRUE, label.pos = "out")
```

`round_df` *Applies round function only on numeric columns of a data.frame.*

Description

Applies round function only on numeric columns of a data.frame.

Usage

```
round_df(df, digits)
```

Arguments

`df`	data.frame object
`digits`	passed to `round`

Value

data.frame object

Examples

```
data("iris")
iris %>% round_df(digits = 0) %>% head()
```

`sampleAssay_plot` *Useful to visualize distribution of assay values for each sample. Plots 'boxplot' of any assay for each sample. Aesthetic can be added from colData.*

Description

Useful to visualize distribution of assay values for each sample. Plots 'boxplot' of any assay for each sample. Aesthetic can be added from colData.

Usage

```
sampleAssay_plot(se, assayName = "counts", ...)
```

Arguments

`se`	Object of `SummarizedExperiment` class
`assayName`	One of the values from SummarizedExperiment::assayNames(se)
`...`	other arguments to be passed to ggpubr::`ggboxplot`

Value

ggplot object

Examples

```
se <- readRDS(system.file("extdata","rat_vole_mouseSE_salmon.rds", package = "broadSeq"))

sampleAssay_plot(se, assayName = "counts",
fill="stage", # stage is a column name of colData(se)
yscale="log2")

se <- broadSeq::normalizeEdgerCPM(se ,method = "none",cpm.log = TRUE )

sampleAssay_plot(se, assayName = "logCPM", fill="stage")
```

`transformDESeq2` *Transform SummarizedExperiment with DESeq2 package*

Description

To use SummarizedExperiment with DESeq2, this function makes sure that 'counts' assay should be the first in assays list and the mode is integer.

Usage

```
transformDESeq2(se, method = "vst", ...)
```

Arguments

`se`	Object of `SummarizedExperiment` class
`method`	"vst", "normTransform" or "rlog" to choose either of DESeq2::`varianceStabilizingTransformation` DESeq2::`normTransform` and DESeq2::`rlog` function. default is "vst"
`...`	arguments passed to `varianceStabilizingTransformation` `normTransform` and `rlog`

Value

Object of `SummarizedExperiment` class where a new assay is added to the input object.

Examples

```
 se <- readRDS(system.file("extdata",
        "rat_vole_mouseSE_salmon.rds",
        package = "broadSeq"))

se <- broadSeq::transformDESeq2(se,method = "vst"  )
# The transformed values are added with the assay name "vst"
SummarizedExperiment::assayNames(se)
```

`use_DELocal` *To use SummarizedExperiment with package DELocal*

Description

A wrapper function of DELocal where input is an object of `SummarizedExperiment`

Usage

```
use_DELocal(se, colData_id, control, treatment, rank = FALSE, ...)
```

Arguments

`se`	Object of `SummarizedExperiment` class
`colData_id`	One of the columns of colData(se). It should be factors of more than one value.
`control`	Base level and one of the factor values of `colData(se)[[colData_id]]`
`treatment`	one of the factor values of `colData(se)[[colData_id]]`
`rank`	Logical value default FALSE. If true the result will have an additional column named "rank" and the results are ranked on "relative.logFC"
`...`	other arguments to be passed to main function DELocal::`DELocal`.

Value

a data.frame from DELocal

Examples

```
se <- readRDS(system.file("extdata",
        "rat_vole_mouseSE_salmon.rds",
        package = "broadSeq"))

# To reduce runtime
se <- se[rowData(se)$chromosome_name == 2,colData(se)$species == "Mouse"]

result <-
    use_DELocal(se = se,
           colData_id = "stage", control = "Bud", treatment = "Cap",
           rank = TRUE)
```

`use_deseq2` *To use SummarizedExperiment with package DESeq2*

Description

A wrapper function of DESeq2 where input is an object of `SummarizedExperiment`

Usage

```
use_deseq2(se, colData_id, control, treatment, rank = FALSE, ...)
```

Arguments

`se`	Object of `SummarizedExperiment` class
`colData_id`	One of the columns of colData(se). It should be factors of more than one value.
`control`	Base level and one of the factor values of `colData(se)[[colData_id]]`
`treatment`	one of the factor values of `colData(se)[[colData_id]]`
`rank`	Logical value default FALSE. If true the result will have an additional column named "rank" and the results are ranked on "padj"
`...`	other arguments to be passed to main function DESeq2::`results`.

Value

a data.frame converted from DESeq2::`DESeqResults`

Examples

```
se <- readRDS(system.file("extdata",
        "rat_vole_mouseSE_salmon.rds",
        package = "broadSeq"))

# To reduce runtime
se <- se[rowData(se)$chromosome_name == 2,colData(se)$species == "Mouse"]
```

```
result <-
    use_deseq2(se = se,
           colData_id = "stage", control = "Bud", treatment = "Cap",
           rank = TRUE)
```

---

use_EBSeq *To use SummarizedExperiment with package EBSeq*

Description

A wrapper function of EBSeq where input is an object of `SummarizedExperiment`

Usage

```
use_EBSeq(se, colData_id, control, treatment, rank = FALSE, ...)
```

Arguments

`se`	Object of `SummarizedExperiment` class
`colData_id`	One of the columns of colData(se). It should be factors of more than one value.
`control`	Base level and one of the factor values of `colData(se)[[colData_id]]`
`treatment`	one of the factor values of `colData(se)[[colData_id]]`
`rank`	Logical value default FALSE. If true the result will have an additional column named "rank" and the results are ranked on "PPDE"
`...`	other arguments to be passed to main function EBSeq::`GetDEResults`.

Value

a data.frame object converted from the output of EBSeq::`GetDEResults`.

Examples

```
se <- readRDS(system.file("extdata",
        "rat_vole_mouseSE_salmon.rds",
        package = "broadSeq"))

# To reduce runtime
se <- se[rowData(se)$chromosome_name == 2,colData(se)$species == "Mouse"]

result <-
    use_EBSeq(se = se,
           colData_id = "stage", control = "Bud", treatment = "Cap",
           rank = TRUE)
```

`use_edgeR_GLM` *To use SummarizedExperiment with package edgeR*

Description

A wrapper function of DESeq2 where input is an object of `SummarizedExperiment`

Usage

```
use_edgeR_GLM(se, colData_id, control, treatment, rank = FALSE, ...)

use_edgeR_exact(se, colData_id, control, treatment, rank = FALSE, ...)

use_edgeR(
  se,
  colData_id,
  control,
  treatment,
  rank = FALSE,
  edgeR.n = Inf,
  edgeR.adjust.method = "BH",
  edgeR.sort.by = "PValue",
  option = "GLM",
  ...
)
```

Arguments

`se`	Object of `SummarizedExperiment` class
`colData_id`	One of the columns of colData(se). It should be factors of more than one value.
`control`	Base level and one of the factor values of `colData(se)[[colData_id]]`
`treatment`	one of the factor values of `colData(se)[[colData_id]]`
`rank`	Logical value default FALSE. If true the result will have an additional column named "rank"
`...`	other arguments to be passed to edgeR::`glmLRT` or edgeR::`exactTest`
`edgeR.n`	argument for edgeR::`topTags`
`edgeR.adjust.method`	argument for edgeR::`topTags`
`edgeR.sort.by`	argument for edgeR::`topTags`
`option`	"GLM" or "exact" to indicate to use either edgeR::`glmLRT` or edgeR::`exactTest`

Value

a data.frame of output from edgeR::`topTags`

Examples

```
se <- readRDS(system.file("extdata",
        "rat_vole_mouseSE_salmon.rds",
        package = "broadSeq"))

# To reduce runtime
se <- se[rowData(se)$chromosome_name == 2,colData(se)$species == "Mouse"]

result <-
    use_edgeR(se = se,
           colData_id = "stage", control = "Bud", treatment = "Cap",
           rank = TRUE)
```

`use_limma_trend` *To use SummarizedExperiment with package limma*

Description

A wrapper function of limma where input is an object of `SummarizedExperiment`

Usage

```
use_limma_trend(se, colData_id, control, treatment, rank = FALSE, ...)

use_limma_voom(se, colData_id, control, treatment, rank = FALSE, ...)

use_limma(
  se,
  colData_id,
  control,
  treatment,
  rank = FALSE,
  useVoom = TRUE,
  showPlot = FALSE,
  limma.adjust = "BH",
  limma.sort.by = "p",
  limma.number = Inf,
  ...
)
```

Arguments

`se`	Object of `SummarizedExperiment` class
`colData_id`	One of the columns of colData(se). It should be factors of more than one value.
`control`	Base level and one of the factor values of `colData(se)[[colData_id]]`
`treatment`	one of the factor values of `colData(se)[[colData_id]]`

`rank`	Logical value default FALSE. If true the result will have an additional column named "rank"
`...`	other arguments to be passed to main function edgeR::calcNormFactors .
`useVoom`	whether to use limma::voom or edgeR::cpm
`showPlot`	whether to use limma::plotSA ; default FALSE
`limma.adjust`	argument for limma::topTable
`limma.sort.by`	argument for limma::topTable
`limma.number`	argument for limma::topTable

Value

a data.frame of output from limma::topTable

Examples

```
se <- readRDS(system.file("extdata",
        "rat_vole_mouseSE_salmon.rds",
        package = "broadSeq"))

# To reduce runtime
se <- se[rowData(se)$chromosome_name == 2,colData(se)$species == "Mouse"]
result <-
    use_limma(se = se,
           colData_id = "stage", control = "Bud", treatment = "Cap",
           rank = TRUE)
```

`use_multDE` *To identify differentially expressed genes by multiple methods*

Description

To identify differentially expressed genes by multiple methods

Usage

```
use_multDE(
  deFun_list,
  return.df = FALSE,
  se,
  colData_id,
  control,
  treatment,
  ...
)
```

Arguments

`deFun_list`	a list of function which can perform differential expression analysis
`return.df`	whether to return all results aggregated form of data.frame or a list of results. Default is FALSE
`se`	Object of `SummarizedExperiment` class
`colData_id`	One of the columns of colData(se). It should be factors of more than one value.
`control`	Base level and one of the factor values of `colData(se)[[colData_id]]`
`treatment`	one of the factor values of `colData(se)[[colData_id]]`
`...`	other arguments to be passed to functions listed in deFun_list

Value

a list or data.frame

Examples

```
se <- readRDS(system.file("extdata",
        "rat_vole_mouseSE_salmon.rds",
        package = "broadSeq"))

# To reduce runtime
se <- se[rowData(se)$chromosome_name == 2,colData(se)$species == "Mouse"]

# First define a named list of functions
funs <- list(limma_trend = use_limma_trend, limma_voom = use_limma_voom,
             edgeR_exact = use_edgeR_exact, edgeR_glm = use_edgeR_GLM,
             deseq2 = use_deseq2,
             DELocal = use_DELocal, noiseq = use_NOIseq,
             EBSeq = use_EBSeq)


multi_result <- broadSeq::use_multDE(
    se <- se[rowData(se)$chromosome_name == 2,colData(se)$species == "Mouse"],
    deFun_list = funs, return.df = TRUE,
    colData_id = "stage", control = "Bud", treatment = "Cap",
    rank = TRUE)
```

`use_NOIseq` *Differential expression method for NOISeq*

Description

This is a wrapper function of NOISeq::`noiseqbio` whose input class is ´eSet´ and output class is `Output` which are not widely used. We can use as(se, "ExpressionSet") to get an eSet easily but then it will be hard to refer the treatment and control. The order of factors influence the log fold change sign. To keep it comparable to other methods the `NOISeq::readData()` is used internally.

Usage

```
use_NOIseq(se, colData_id, control, treatment, rank = FALSE, ...)
```

Arguments

`se`	Object of `SummarizedExperiment` class
`colData_id`	One of the columns of colData(se). It should be factors of more than one value.
`control`	Base level and one of the factor values of `colData(se)[[colData_id]]`
`treatment`	one of the factor values of `colData(se)[[colData_id]]`
`rank`	Logical value default FALSE. If true the result will have an additional column named "rank" which is ordered by ´prob´ values returned by function NOISeq::`noiseqbio`.
`...`	other arguments to be passed to main function NOISeq::`noiseqbio`. The 'input' and 'factor' argument should not be used.

Value

A data.frame object from the results of NOISeq::noiseqbio(). For details check the documentation of ´NOISeq´

Examples

```
se <- readRDS(system.file("extdata",
        "rat_vole_mouseSE_salmon.rds",
        package = "broadSeq"))

# To reduce runtime
se <- se[rowData(se)$chromosome_name == 2,colData(se)$species == "Mouse"]

result_Noiseq <-
    use_NOIseq(se = se,
           colData_id = "stage", control = "Bud", treatment = "Cap",
           rank = TRUE,
           r = 10) # r is an argument of NOISeq::noiseqbio
```

`use_SAMseq` *To use SummarizedExperiment with package samr*

Description

To use SummarizedExperiment with package samr

Usage

```
use_SAMseq(se, colData_id, control, treatment, rank = FALSE, ...)
```

Arguments

`se`	Object of `SummarizedExperiment` class
`colData_id`	One of the columns of colData(se). It should be factors of more than one value.
`control`	Base level and one of the factor values of `colData(se)[[colData_id]]`
`treatment`	one of the factor values of `colData(se)[[colData_id]]`
`rank`	Logical value default FALSE. If true the result will have an
`...`	other arguments to be passed to samr::SAMseq

Value

a data.frame object as a result

`volcanoPlot` *Volcano plot with formatted x and y axis label.*

Description

Volcano plot with formatted x and y axis label.

Usage

```
volcanoPlot(
  df,
  pValName,
  lFCName,
  sigThreshold = 0.05,
  logFCThreshold = 1,
  labelName = NULL,
  selectedLabel = NULL,
  palette = "nejm"
)
```

Arguments

`df`	a data.frame object
`pValName`	column name of df which provides p-values
`lFCName`	column name of df which provides log fold change values
`sigThreshold`	Threshold for p-values
`logFCThreshold`	Threshold for log fold change values
`labelName`	column name of df to label the dots
`selectedLabel`	which dots to highlight
`palette`	one of "npg" ,"aaas", "lancet", "jco", "ucscgb", "uchicago", "simpsons" and "nejm" or similar to viridis::cividis(3)

Value

ggplot object

Index