Package ‘escape’

March 22, 2024

Title Easy single cell analysis platform for enrichment
Version 1.12.0
Date 2022-3-05

Description A bridging R package to facilitate gene set enrichment analysis (GSEA) in the context of single-cell RNA sequencing. Using raw count information, Seurat objects, or SingleCellExperiment format, users can perform and visualize GSEA across individual cells.

License GPL-2
Encoding UTF-8
LazyData false
RoxygenNote 7.1.2
biocViews Software, SingleCell, Classification, Annotation, GeneSetEnrichment, Sequencing, GeneSignaling, Pathways
Depends R (>= 4.1)
Imports grDevices, dplyr, ggplot2, GSEABase, GSVA, SingleCellExperiment, ggiradis, mslidgbr, stats, BiocParallel, Matrix, UCell, broom, reshape2, patchwork, MatrixGenerics, utils, rlang, stringr, data.table, SummarizedExperiment, methods
Suggests Seurat, SeuratObject, knitr, rmarkdown, markdown, BiocStyle, testthat, dittoSeq (>= 1.1.2)
VignetteBuilder knitr
git_url https://git.bioconductor.org/packages/escape
git_branch RELEASE_3_18
git_last_commit 51adca6
git_last_commit_date 2023-10-24
Repository Bioconductor 3.18
Date/Publication 2024-03-22

Author Nick Borcherding [aut, cre], Jared Andrews [aut]
Maintainer Nick Borcherding <ncborch@gmail.com>
calculate_Uscore

R topics documented:

calculate_Uscore .......................... 2
check_genes ............................. 3
check_signature_names .................. 4
data_to_ranks_data_table .............. 4
enrichIt .................................. 5
enrichmentPlot ........................... 6
escape_gene_sets ........................ 7
geneSets .................................. 7
getGeneSets ............................... 8
getSignificance ............................ 8
masterPCAPlot ............................. 9
pcaEnrichment ............................. 10
performPCA ............................... 11
rankings2Uscore ........................... 11
ridgeEnrichment .......................... 12
ScoreSignatures_UCell ................... 13
splitEnrichment ........................... 14
StoreRankings_UCell ..................... 15
u_stat ..................................... 16
u_stat_signature_list .................... 17

Index ................................. 18

|
| calculate_Uscore | Calculate rankings and scores for query data and given signature set |

**Description**

Calculate rankings and scores for query data and given signature set

**Usage**

```r
calculate_Uscore(
  matrix,
  features,
  maxRank = 1500,
  chunk.size = 1000,
  ncores = 1,
  w_neg = 1,
  ties.method = "average",
  storeRanks = FALSE,
  force.gc = FALSE,
  name = ""
)
```
**Arguments**

- **matrix**: Input data matrix
- **features**: List of signatures
- **maxRank**: Rank cutoff (1500)
- **chunk.size**: Cells per sub-matrix (1000)
- **ncores**: Number of cores to use for parallelization (1)
- **w_neg**: Weight on negative signatures
- **ties.method**: How to break ties, for data.table::frankv method ("average")
- **storeRanks**: Store ranks? (FALSE)
- **force.gc**: Force garbage collection? (FALSE)
- **name**: Suffix for metadata columns ("_UCell")

**Value**

A list of signature scores

---

**check_genes**

`Check if all genes in signatures are found in data matrix - otherwise add zero counts in data-matrix to complete it`

**Description**

Check if all genes in signatures are found in data matrix - otherwise add zero counts in data-matrix to complete it

**Usage**

`check_genes(matrix, features)`

**Arguments**

- **matrix**: Input data matrix
- **features**: List of genes that must be present (otherwise they are added)

**Value**

Same input matrix, extended to comprise any missing genes
check_signature_names  Check signature names and add standard names is missing

Description
Check signature names and add standard names is missing

Usage
check_signature_names(features)

Arguments
features      List of signatures for scoring

Value
The input list of signatures, with standard names if provided un-named

data_to_ranks_data_table
Calculate per-cell feature rankings

Description
Calculate per-cell feature rankings

Usage
data_to_ranks_data_table(data, ties.method = "average")

Arguments
data        Expression data matrix
ties.method  How to break ties (passed on to data.table::frankv)

Value
A data.table of ranks
Description

This function allows users to input both the single-cell RNA-sequencing counts and any gene set pathways either from the stored data or from other sources. The enrichment calculation itself uses the two methods 1) gs va R package and the poisson distribution for RNA or the UCell package.

Usage

enrichIt(
  obj,
  gene.sets = NULL,
  method = "ssGSEA",
  groups = 1000,
  cores = 2,
  min.size = 5,
  ssGSEA.norm = FALSE,
  ...
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>obj</td>
<td>The count matrix, Seurat, or SingleCellExperiment object.</td>
</tr>
<tr>
<td>gene.sets</td>
<td>Gene sets from getGeneSets to use for the enrichment analysis. Alternatively a simple base R list where the names of the list elements correspond to the name of the gene set and the elements themselves are simple vectors of gene names representing the gene set.</td>
</tr>
<tr>
<td>method</td>
<td>select the method to calculate enrichment, either &quot;ssGSEA&quot; or &quot;UCell&quot;</td>
</tr>
<tr>
<td>groups</td>
<td>The number of cells to separate the enrichment calculation.</td>
</tr>
<tr>
<td>cores</td>
<td>The number of cores to use for parallelization.</td>
</tr>
<tr>
<td>min.size</td>
<td>Minimum number of gene necessary to perform the enrichment calculation</td>
</tr>
<tr>
<td>ssGSEA.norm</td>
<td>normalized the enrichment score based on the range of the individual gene set. If TRUE, the returned enrichment score is based may change with cell composition.</td>
</tr>
</tbody>
</table>

Value

Data frame of normalized enrichment scores (NES)

Author(s)

Nick Borcherding, Jared Andrews
enrichmentPlot

See Also

geneSets to collect gene sets.

Examples

GS <- list(Bcells = c("MS4A1", "CD79B", "CD79A", "IGH1", "IGH2"),
            Tcells = c("CD3E", "CD3D", "CD3G", "CD7", "CD8A"))
pbmc_small <- suppressWarnings(SeuratObject::pbmc_small)
ES <- enrichIt(obj = pbmc_small, gene.sets = GS, min.size = NULL)

enrichmentPlot

Description

Gene Rank Enrichment Plot Display the rank order density for indi-
vidual gene sets by group identify. This function will use the group
variable to take the mean rank order across all individual cells.

Usage

enrichmentPlot(
  obj,
  gene.set,
  gene.sets,
  group,
  colors = c("#0D0887FF", "#7E03A8FF", "#CC4678FF", "#F89441FF", "#F0F921FF")
)

Arguments

obj The Seurat or SingleCellExperiment object.
gene.set The name of the specific gene set to visualize
gene.sets Gene sets from geneSets to use
group The header in the meta data that will be used for the comparison
colors The color palette for the enrichment plot

Value

ggplot2 object mean rank gene density
### Examples

```r
## Not run:
GS <- list(Housekeeping = c("ACTA1", "ACTN1", "GAPDH"),
            Cancer = c("TP53", "BRCA2", "ERBB2", "MYC"))
pbm_small <- suppressWarnings(SeuratObject::pbmc_small)

enrichmentPlot(pbm_small gene.set = "Cancer",
                gene.sets = GS, group = "groups")

## End(Not run)
```

---

**escape.gene.sets**  
***In-Built Gene Sets for Escape***

### Description

A list of gene sets derived from PMID: 29961579 relating to tumor immunity.

---

**getGeneSets**  
***Get a collection of gene sets to perform enrichment on***

### Description

This function allows users to select libraries and specific gene.sets to form a GeneSetCollection that is a list of gene sets.

### Usage

```r
getGeneSets(
  species = "Homo sapiens",
  library = NULL,
  subcategory = NULL,
  gene.sets = NULL
)
```

### Arguments

- **species**: The scientific name of the species of interest in order to get correcr gene nomenclature
- **library**: Individual collection(s) of gene sets, e.g. c("H", "C5"). See msigdb for all MSigDB collections.
- **subcategory**: MSigDB sub-collection abbreviation, such as CGP or BP.
- **gene.sets**: Select gene sets or pathways, using specific names, example: pathways = c("HALLMARK_TNFA_SIGNALING_VIA_NFKB")
  Will only be honored if library is set, too.
getSignificance

Value
A GeneSetCollection object containing the requested GeneSet objects.

Author(s)
Nick Borcherding, Jared Andrews

Examples
GS <- getGeneSets(library = "H")

---

getSignificance  
Perform significance testing between groups and enrichment scores.

Description
This function takes the enrichment scores and performs statistical testing to evaluate the difference by group selected. The function can perform 5 tests: 1) Welch’s T test (T.test), 2) Logistic Regression (LR), 3) Wilcoxon Rank Sum Test (Wilcoxon), 4) one-way ANOVA (ANOVA), and 5) Kruskal-Wallis (KW). The latter two output will include the individual comparisons between groups using TukeyHSD for ANOVA and pairwise Wilcoxon Rank Sum Test for KW. The output includes adjusted p-values based on the Benjamini Hochberg method.

Usage
getSignificance(enriched, group = NULL, gene.sets = NULL, fit = NULL)

Arguments
- enriched: The output of enrichIt.
- group: The parameter to group for the comparison, should a column of the enriched input
- gene.sets: Names of gene sets to compare
- fit: The test used for significance, 2 group: Wilcoxon, LR, T.test. Multigroup: ANOVA or KW.

Value
Data frame of test statistics

See Also
enrichIt for generating enrichment scores.
masterPCAPlot

Examples

ES2 <- readRDS(url("https://ncborcherding.github.io/vignettes/escape_enrichment_results.rds"))
output <- getSignificance(ES2, group = "Type", fit = "T.test")

masterPCAPlot(enriched, gene.sets, PCx, PCy, top.contribution = 10)

masterPCAPlot

Visualize the components of the PCA analysis of the enrichment results

Description

Graph the major gene set contributors to the pcaEnrichment.

Usage

masterPCAPlot(enriched, gene.sets, PCx, PCy, top.contribution = 10)

Arguments

enriched The output of enrichIt.
gene.sets Names of gene sets to include in the PCA
PCx The principal component graphed on the x-axis.
PCy The principal component graphed on the y-axis.
top.contribution The number of gene sets to graph, organized by PCA contribution.

Value

ggplot2 object summarizing the PCA for the enrichment scores

See Also

enrichIt for generating enrichment scores.

Examples

ES2 <- readRDS(url("https://ncborcherding.github.io/vignettes/escape_enrichment_results.rds"))

masterPCAPlot(ES2, PCx = "PC1", PCy = "PC2", gene.sets = colnames(ES2),
top.contribution = 10)
**pcaEnrichment**

Density plot of the principal components

### Description

Density plot of the principal components

### Usage

```r
pcaEnrichment(
  PCAout,
  PCx,
  PCy,
  colors = c("#0D0887FF", "#7E03A8FF", "#CC4678FF", "#F89441FF", "#F0F921FF"),
  contours = TRUE,
  facet = NULL
)
```

### Arguments

- **PCAout**: The output of `performPCA`
- **PCx**: The principal component graphed on the x-axis
- **PCy**: The principal component graphed on the y-axis
- **colors**: The color palette for the density plot
- **contours**: Binary classifier to add contours to the density plot
- **facet**: A parameter to separate the graph

### Value

A `ggplot2` object of the results of PCA for the enrichment scores

### See Also

`performPCA` for generating PCA results.

### Examples

```r
ES2 <- readRDS(url(  
  "https://ncborcherding.github.io/vignettes/escape_enrichment_results.rds"))
PCA <- performPCA(enriched = ES2, groups = c("Type", "Cluster"))
pcaEnrichment(PCA, PCx = "PC1", PCy = "PC2", contours = TRUE)
```
Perform PCA

**Calculate Principal Components for the Enrichment Scores**

### Description

Using all or selected enrichment scores of individual single-cells, this function will calculate principal components using scaled values and attach to the output columns to use to graph later.

### Usage

```r
performPCA(enriched, gene.sets = NULL, groups)
```

### Arguments

- **enriched**: The output of `enrichIt`.
- **gene.sets**: Names of gene sets to include in the PCA.
- **groups**: The column headers to use in future graphing functions.

### Value

Data frame of principal components.

### Author(s)

Nick Borcherding

### Examples

```r
ES2 <- readRDS(url(  
  "https://ncborcherding.github.io/vignettes/escape_enrichment_results.rds"))
PCA <- performPCA(enriched = ES2, groups = c("Type", "Cluster"),  
gene.sets = colnames(ES2))
```

---

Rankings2Uscore

**Get signature scores from pre-computed rank matrix**

### Description

Get signature scores from pre-computed rank matrix.
Usage

rankings2Uscore(
    ranks_matrix,
    features,
    chunk.size = 1000,
    w_neg = 1,
    ncores = 1,
    force.gc = FALSE,
    name = "_UCell"
)

Arguments

ranks_matrix A rank matrix
features List of signatures
chunk.size How many cells per matrix chunk
w_neg Weight on negative signatures
ncores How many cores to use for parallelization
force.gc Force garbage collection to recover RAM? (FALSE)
name Name suffix for metadata columns ("_UCell")

Value

A list of signature scores

---

ridgeEnrichment Generate a ridge plot to examine enrichment distributions

Description

This function allows the user to examine the distribution of enrichment across groups by generating a ridge plot.

Usage

ridgeEnrichment(
    enriched,
    group = "cluster",
    gene.set = NULL,
    scale.bracket = NULL,
    facet = NULL,
    add.rug = FALSE,
    colors = c("#0D0887FF", "#7E03A8FF", "#CC4678FF", "#F89441FF", "#F0F921FF")
)
ScoreSignatures_UCell

Arguments

- **enriched**: The output of `enrichIt`
- **group**: The parameter to group, displayed on the y-axis.
- **gene.set**: The gene set to graph on the x-axis.
- **scale.bracket**: This will filter the enrichment scores to remove extreme outliers. Values entered (1 or 2 numbers) will be the filtering parameter using z-scores of the selected gene.set. If only 1 value is given, a secondary bracket is automatically selected as the inverse of the number.
- **facet**: A parameter to separate the graph.
- **add.rug**: Binary classifier to add a rug plot to the x-axis.
- **colors**: The color palette for the ridge plot.

Value

ggplot2 object with ridge-based distributions of selected gene.set

See Also

`enrichIt` for generating enrichment scores.

Examples

```r
ES2 <- readRDS(url(
  "https://ncborcherding.github.io/vignettes/escape_enrichment_results.rds"))
ridgeEnrichment(ES2, gene.set = "HALLMARK_DNA_REPAIR", group = "cluster",
facet = "Type", add.rug = TRUE)
```

ScoreSignatures_UCell  Calculate module enrichment scores from single-cell data

Description

Calculate module enrichment scores from single-cell data

Usage

```r
ScoreSignatures_UCell(
  matrix = NULL,
  features,
  precalc.ranks = NULL,
  maxRank = 1500,
  w_neg = 1,
  name = ".UCell",
  assay = "counts",
  chunk.size = 1000,
)```
splitEnrichment

Arguments

matrix Input matrix,
features A list of signatures
precalc.ranks If you have pre-calculated ranks
maxRank Maximum number of genes to rank per cell; above this rank, a given gene is considered as not expressed.
w_neg Weight on negative genes in signature. e.g. ‘w_neg=1’ weighs equally up- and down-regulated genes.
name Name suffix appended to signature names
assay The sce object assay where the data is to be found
chunk.size Number of cells to be processed simultaneously (lower size requires slightly more computation but reduces memory demands)
ncores Number of processors to parallelize computation.
ties.method How ranking ties should be resolved (passed on to [data.table::frank])
force.gc Explicitly call garbage collector to reduce memory footprint
seed Integer seed

Value

Returns input SingleCellExperiment object with UCell scores added to altExp

Description

This function allows to the user to examine the distribution of enrichment across groups by generating a split violin plot.

Usage

splitEnrichment(
enriched,
x.axis = NULL,
scale.bracket = NULL,
split = NULL,
gene.set = NULL,
colors = c("#0D0887FF", "#7E03A8FF", "#CC4678FF", "#F89441FF", "#F0F921FF")
)
Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>enriched</td>
<td>The output of <code>enrichIt</code></td>
</tr>
<tr>
<td>x.axis</td>
<td>Optional parameter for separation.</td>
</tr>
<tr>
<td>scale.bracket</td>
<td>This will filter the enrichment scores to remove extreme outliers. Values entered (1 or 2 numbers) will be the filtering parameter using z-scores of the selected gene.set. If only 1 value is given, a secondary bracket is automatically selected as the inverse of the number.</td>
</tr>
<tr>
<td>split</td>
<td>The parameter to split, must be binary.</td>
</tr>
<tr>
<td>gene.set</td>
<td>The gene set to graph on the y-axis.</td>
</tr>
<tr>
<td>colors</td>
<td>The color palette for the ridge plot.</td>
</tr>
</tbody>
</table>

Value

`ggplot2` object violin-based distributions of selected gene.set

See Also

`enrichIt` for generating enrichment scores.

Examples

```r
ES2 <- readRDS(url("https://ncborcherding.github.io/vignettes/escape_enrichment_results.rds"))
splitEnrichment(ES2, x.axis = "cluster", split = "Type", gene.set = "HALLMARK_DNA_REPAIR")
```

---

**StoreRankings_UCell**  
*Calculate and store gene rankings for a single-cell dataset*

Description

Calculate and store gene rankings for a single-cell dataset

Usage

```r
StoreRankings_UCell(
  matrix,
  maxRank = 1500,
  chunk.size = 1000,
  ncores = 1,
  assay = "counts",
  ties.method = "average",
  force.gc = FALSE,
  seed = 123
)
```
u_stat

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>matrix</td>
<td>Input matrix, either stored in a [SingleCellExperiment] object or as a raw matrix. dgCMatrix format supported.</td>
</tr>
<tr>
<td>maxRank</td>
<td>Maximum number of genes to rank per cell; above this rank, a given gene is considered as not expressed</td>
</tr>
<tr>
<td>chunk.size</td>
<td>Number of cells to be processed simultaneously (lower size requires slightly more computation but reduces memory demands)</td>
</tr>
<tr>
<td>ncores</td>
<td>Number of processors to parallelize computation</td>
</tr>
<tr>
<td>assay</td>
<td>Assay where the data is to be found (for input in 'sce' format)</td>
</tr>
<tr>
<td>ties.method</td>
<td>How ranking ties should be resolved (passed on to [data.table::frank])</td>
</tr>
<tr>
<td>force.gc</td>
<td>Explicitly call garbage collector to reduce memory footprint</td>
</tr>
<tr>
<td>seed</td>
<td>Integer seed</td>
</tr>
</tbody>
</table>

Value

- Returns a sparse matrix of pre-calculated ranks that can be used multiple times to evaluate different signatures
- Returns a sparse matrix of pre-calculated ranks that can be used multiple times to evaluate different signatures

u_stat

Calculate Mann Whitney U from a vector of ranks

Description

Calculate Mann Whitney U from a vector of ranks

Usage

u_stat(rank_value, maxRank = 1000, sparse = FALSE)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rank_value</td>
<td>A vector of ranks</td>
</tr>
<tr>
<td>maxRank</td>
<td>Max number of features to include in ranking</td>
</tr>
<tr>
<td>sparse</td>
<td>Whether the vector of ranks is in sparse format</td>
</tr>
</tbody>
</table>

Value

Normalized AUC (as U statistic) for the vector
u_stat_signature_list  Calculate U scores for a list of signatures, given a rank matrix

Description
Calculate U scores for a list of signatures, given a rank matrix

Usage
u_stat_signature_list(
    sig_list,
    ranks_matrix,
    maxRank = 1000,
    sparse = FALSE,
    w_neg = 1
)

Arguments

- **sig_list**: A list of signatures
- **ranks_matrix**: Matrix of pre-computed ranks
- **maxRank**: Max number of features to include in ranking, for u_stat function
- **sparse**: Whether the vector of ranks is in sparse format
- **w_neg**: Weight on negative signatures

Value
A matrix of U scores
Index

calculate_Uscore, 2
check_genes, 3
check_signature_names, 4
data_to_ranks_data_table, 4
enrichIt, 5, 8, 9, 11, 13, 15
enrichmentPlot, 6
escape.gene.sets, 7
getGeneSets, 5, 6, 7
getSignificance, 8
masterPCAPlot, 9
pcaEnrichment, 9, 10
performPCA, 10, 11
rankings2Uscore, 11
ridgeEnrichment, 12
ScoreSignatures_UCell, 13
splitEnrichment, 14
StoreRankings_UCell, 15
u_stat, 16
u_stat_signature_list, 17