Package 'escape'

July 16, 2025

Title Easy single cell analysis platform for enrichment

Version 2.5.4

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 ssGSEA, GSVA, AUCell, and UCell-based enrichment calculations across individual cells. Alternatively, escape supports use of rank-based GSEA, such as the use of differential gene expression via fgsea.

License MIT + file LICENSE

Encoding UTF-8

LazyData false

RoxygenNote 7.3.2

biocViews Software, SingleCell, Classification, Annotation, GeneSetEnrichment, Sequencing, GeneSignaling, Pathways

Depends R (>= 4.1)

Imports ggdist, ggplot2 (>= 3.5.0), grDevices, Matrix, MatrixGenerics, methods, stats, SummarizedExperiment, utils

Suggests AUCell, BiocParallel, BiocStyle, DelayedMatrixStats, dplyr, fgsea, GSEABase, ggraph, ggridges, ggpointdensity, GSVA, hexbin, igraph, irlba, knitr, msigdb, patchwork, rmarkdown, rlang, scran, SeuratObject, Seurat, SingleCellExperiment, spelling, stringr, testthat (>= 3.0.0), UCell

VignetteBuilder knitr

Language en-US

BugReports https://github.com/BorchLab/escape/issues

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

git_branch devel

git_last_commit b013e24

git_last_commit_date 2025-06-05

Repository Bioconductor 3.22

Date/Publication 2025-07-16

Author Nick Borcherding [aut, cre], Jared Andrews [aut], Tobias Hoch [ctb], Alexei Martsinkovskiy [ctb]

Maintainer Nick Borcherding <ncborch@gmail.com>

Contents

densityEnrichment . . . 2
enrichIt . . . 3
enrichItPlot . . . 4
escape.gene.sets . . . 5
escape.matrix . . . 6
getGeneSets . . . 8
geyserEnrichment . . . 9
gseaEnrichment . . . 10
heatmapEnrichment . . . 11
pcaEnrichment . . . 12
performNormalization . . . 14
performPCA . . . 15
ridgeEnrichment . . . 16
runEscape . . . 17
scatterEnrichment . . . 19
splitEnrichment . . . 20

Index **22**

`densityEnrichment` *Visualize Mean Density Ranking of Genes Across Gene Sets*

Description

This function allows to the user to examine the mean ranking within the groups across the gene set. The visualization uses the density function to display the relative position and distribution of rank.

Usage

```
densityEnrichment(
  input.data,
  gene.set.use,
  gene.sets,
  group.by = NULL,
  rug.height = 0.02,
  palette = "inferno"
)
```

Arguments

`input.data`	A Seurat object or a SingleCellExperiment.
`gene.set.use`	Character(1). Name of the gene set to display.
`gene.sets`	A named list of character vectors, the result of [getGeneSets()], or the built-in data object [escape.gene.sets].
`group.by`	Metadata column. Defaults to the Seurat/SCE ‘ident‘ slot when ‘NULL‘.
`rug.height`	Vertical spacing of the hit rug as a fraction of the y-axis (default ‘0.02‘).
`palette`	Character. Any palette from `hcl.pals`.

Value

A ‘patchwork‘/‘ggplot2‘ object.

Examples

```
gs <- list(Bcells = c("MS4A1", "CD79B", "CD79A", "IGH1", "IGH2"),
           Tcells = c("CD3E", "CD3D", "CD3G", "CD7","CD8A"))

pbmc_small <- SeuratObject::pbmc_small

densityEnrichment(pbmc_small,
                  gene.set.use = "Tcells",
                  gene.sets = gs)
```

`enrichIt` *Flexible GSEA for Precomputed Gene Lists*

Description

A convenience front-end to **fgsea** that lets you point at the ‘avg_log2FC‘ and ‘p_val_adj‘ columns coming out of Seurat / DESeq2 / edgeR etc. It converts them to a signed -log10(*p*) ranking, filters on significance / effect size, and then runs fgsea.

Usage

```
enrichIt(
  input.data,
  gene.sets,
  gene_col = NULL,
  logFC_col = "avg_log2FC",
  pval_col = c("p_val_adj", "p_val"),
  ranking_fun = c("signed_log10_p", "logFC"),
  pval_cutoff = 1,
  logFC_cutoff = 0,
  minSize = 5,
  maxSize = 500,
  padjust_method = "BH",
  nproc = 0
)
```

Arguments

`input.data`	Either • a named numeric vector **already ranked**, *or* • a data.frame/tibble with one row per gene and columns containing log-fold-change and *p*-value. If the gene ID is not in ‘rownames(data)‘, supply ‘gene_col‘.
`gene.sets`	AA named list of character vectors, the result of [getGeneSets()], or the built-in data object [escape.gene.sets].
`gene_col`	Name of the column holding gene identifiers (ignored when they are row-names). Default ‘NULL‘.

`logFC_col`, `pval_col`
: Column names for logFC and *p* (or adj.*p*) – defaults match Seurat's 'FindMarkers()'.

`ranking_fun`
: How to build the ranking: '"signed_log10_p"' (default) or '"logFC"'.

`pval_cutoff`, `logFC_cutoff`
: Filters applied **before** ranking.

`minSize`, `maxSize`
: Integer. Minimum / maximum pathway size passed to *fgsea* (default 5 / 500).

`padjust_method`
: Multiple-testing correction; any method accepted by [stats::p.adjust()] (default '"BH"').

`nproc`
: Passed to **fgsea** ('0' = multithread if OpenMP available).

Value

'data.frame' with the usual fgsea columns plus a convenient 'leadingEdge' character column collapsed with \";\".

See Also

[fgsea::fgsea()], [getGeneSets()], [gseaEnrichment()]

Examples

```
pbmc_small <- SeuratObject::pbmc_small

Seurat::Idents(pbmc_small) <- "groups"
markers <- Seurat::FindMarkers(pbmc_small,
                               ident.1 = "g1",
                               ident.2 = "g2")

gs <- list(Bcells = c("MS4A1", "CD79B", "CD79A", "IGH1", "IGH2"),
           Tcells = c("CD3E", "CD3D", "CD3G", "CD7","CD8A"))

gsea <- enrichIt(markers,
                 gene.sets = gs)
```

---

enrichItPlot *Adaptive visualisation of enrichIt results*

Description

Adaptive visualisation of enrichIt results

Usage

```
enrichItPlot(
  res,
  plot.type = c("bar", "dot", "cnet"),
  top = 20,
  x.measure = "-log10(padj)",
```

```
  color.measure = x.measure,
  show.counts = TRUE,
  palette = "inferno",
  ...
)
```

Arguments

res	'data.frame' returned by [enrichIt()].
plot.type	'"bar"', '"dot"', or '"cnet"'.
top	Integer. Keep the top *n* terms **per database** (ranked by adjusted *p*). Set to 'Inf' to keep all.
x.measure	A column in 'res' mapped to the *x*-axis (ignored for '"cnet"'). Default '"-log10(padj)"'.
color.measure	Column mapped to color (dot plot only). Default same as 'x.measure'.
show.counts	Logical. Annotate bar plot with the 'Count' (number of genes).
palette	palette Character. Any palette from hcl.pals.
...	Further arguments passed to **ggplot2** geoms (e.g. 'alpha', 'linewidth').

Value

A **patchwork** object (bar / dot) or **ggraph** object (cnet).

Examples

```
## Not run:
ranks <- setNames(markers$avg_log2FC, rownames(markers))
gs    <- getGeneSets("Homo sapiens", library = c("H", "C2"))
res   <- enrichIt(ranks, gs)

enrichItPlot(res)
enrichItPlot(res, "dot", top=10)
enrichItPlot(res, "cnet", top=5)

## End(Not run)
```

---

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

Description

'escape.gene.sets' ships with **escape** and provides a convenient set of cell-type and pathway signatures from the scRNA-seq tumor micro-environment study by Azizi *et al.* (2018, Cell doi:10.1016/j.cell.2018.06.021). These signatures capture major immune and stromal populations observed across breast-cancer samples and serve as a lightweight default for quick testing or exploratory analyses.

Usage

```
data("escape.gene.sets")
```

Details

The original paper defined cell-type signatures as the top differentially expressed genes per cluster (Azizi *et al.*, Supplementary Table S3).

Source

Supplementary Table S3 in Azizi *et al.* (2018) <https://pubmed.ncbi.nlm.nih.gov/29961579/>

References

Azizi E, *et al.* **Single-cell map of diverse immune phenotypes in the breast tumor microenvironment.** *Cell* 173(5):1293-1308 (2018).

See Also

[runEscape()], [escape.matrix()], [getGeneSets()]

`escape.matrix` *Calculate Single-Cell Gene-Set Enrichment Scores*

Description

‘escape.matrix()‘ computes per-cell enrichment for arbitrary gene-set collections using one of four scoring back-ends and returns a dense numeric matrix (cells × gene-sets). The expression matrix is processed in user-defined *chunks* (‘groups‘) so that memory use remains predictable; each chunk is dispatched in parallel via a **BiocParallel** ‘BPPARAM‘ backend. Heavy engines (**GSVA**, **UCell**, **AUCell**) are loaded lazily, keeping them in the package’s **Suggests** field.

Usage

```
escape.matrix(
  input.data,
  gene.sets = NULL,
  method = "ssGSEA",
  groups = 1000,
  min.size = 5,
  normalize = FALSE,
  make.positive = FALSE,
  min.expr.cells = 0,
  min.filter.by = NULL,
  BPPARAM = NULL,
  ...
)
```

Arguments

`input.data`	A raw-counts matrix (‘genes × cells‘), a Seurat object, or a SingleCellExperiment. Gene identifiers must match those in ‘gene.sets‘.
`gene.sets`	A named list of character vectors, the result of [getGeneSets()], or the built-in data object [escape.gene.sets]. List names become column names in the result.

method	Scoring algorithm (case-insensitive). One of '"GSVA"', '"ssGSEA"', '"UCell"', or '"AUCell"'. Default **'"ssGSEA"'**.
groups	Integer >= 1. Number of cells per processing chunk. Larger values reduce overhead but increase memory usage. Default **1000**.
min.size	Minimum number of genes from a set that must be detected in the expression matrix for that set to be scored. Default **5**. Use 'NULL' to disable filtering.
normalize	Logical. If 'TRUE', the score matrix is passed to [performNormalization()] (drop-out scaling and optional log transform). Default **FALSE**.
make.positive	Logical. If 'TRUE' *and* 'normalize = TRUE', shifts every gene-set column so its global minimum is zero, facilitating downstream log-ratio analyses. Default **FALSE**.
min.expr.cells	Numeric. Gene-expression filter threshold (see details above). Default **0** (no gene filtering).
min.filter.by	Character or 'NULL'. Column name in 'meta.data' (Seurat) or 'colData' (SCE) defining groups within which the 'min.expr.cells' rule is applied. Default **'NULL'**.
BPPARAM	A **BiocParallel** parameter object describing the parallel backend.
...	Extra arguments passed verbatim to the chosen back-end scoring function ('gsva()', 'ScoreSignatures_UCell()', or 'AUCell_calcAUC()').

Value

A numeric matrix with one row per cell and one column per gene set, ordered as in 'gene.sets'.

Supported methods

'"GSVA"' Gene-set variation analysis (Poisson kernel).

'"ssGSEA"' Single-sample GSEA.

'"UCell"' Rank-based UCell scoring.

'"AUCell"' Area-under-the-curve ranking score.

Author(s)

Nick Borcherding, Jared Andrews

See Also

[runEscape()] to attach scores to a single-cell object; [getGeneSets()] for MSigDB retrieval; [performNormalization()] for the optional normalization workflow.

Examples

```
gs <- list(Bcells = c("MS4A1", "CD79B", "CD79A", "IGH1", "IGH2"),
           Tcells = c("CD3E", "CD3D", "CD3G", "CD7","CD8A"))

pbmc <- SeuratObject::pbmc_small
es   <- escape.matrix(pbmc,
                      gene.sets = gs,
                      method = "ssGSEA",
                      groups = 500,
                      min.size = 3)
```

getGeneSets *Get a collection of gene sets from the msigdb*

Description

This function retrieves gene sets from msigdb and caches the downloaded object for future calls. It allows subsetting by main collection (library), subcollection, or specific gene sets, and only supports human ("Homo sapiens") and mouse ("Mus musculus").

Usage

```
getGeneSets(
  species = c("Homo sapiens", "Mus musculus"),
  library = NULL,
  subcategory = NULL,
  gene.sets = NULL,
  version = "7.4",
  id = "SYM"
)
```

Arguments

species	‘"Homo sapiens"‘ (default) or ‘"Mus musculus"‘.
library	Character. Optional vector of main collection codes (e.g. ‘"H"‘, ‘"C5"‘).
subcategory	Character. Optional vector of sub-collection codes (e.g. ‘"GO:BP"‘).
gene.sets	Character. Optional vector of specific gene-set names.
version	MSigDB version (character, default ‘"7.4"‘).
id	Identifier type (default ‘"SYM"‘ for symbols).

Value

A named ‘list‘ of character vectors (gene IDs).

Examples

```
## Not run:
# Get all hallmark gene sets from human.
gs <- getGeneSets(species = "Homo sapiens",
                  library = "H")

# Get a subset based on main collection and subcollection.
gs <- getGeneSets(species = "Homo sapiens",
                  library = c("C2", "C5"),
                  subcategory = "GO:BP")

## End(Not run)
```

`geyserEnrichment` *Visualize Enrichment Distributions Using Geyser Plots*

Description

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

Usage

```
geyserEnrichment(
  input.data,
  assay = NULL,
  group.by = NULL,
  gene.set,
  color.by = "group",
  order.by = NULL,
  scale = FALSE,
  facet.by = NULL,
  palette = "inferno"
)
```

Arguments

`input.data`	Output of `escape.matrix` or a single-cell object previously processed by `runEscape`.
`assay`	Name of the assay holding enrichment scores when 'input.data' is a single-cell object. Ignored otherwise.
`group.by`	Metadata column plotted on the *x*-axis. Defaults to the Seurat/SCE 'ident' slot when 'NULL'.
`gene.set`	Character(1). Gene-set to plot (must exist in the enrichment matrix).
`color.by`	Aesthetic mapped to point color. Use either *"group"* (default = 'group.by') for categorical coloring or the *name of a gene-set* (e.g. same as 'gene.set') to obtain a numeric accepted.
`order.by`	How to arrange the x-axis: *'"mean"'* – groups ordered by decreasing group mean; *'"group"'* – natural sort of group labels; *'NULL'* – keep original ordering.
`scale`	Logical; if 'TRUE' scores are centered/scaled (Z-score) prior to plotting.
`facet.by`	Optional metadata column used to facet the plot.
`palette`	Character. Any palette from `hcl.pals`.

Value

A **ggplot2** object.

Examples

```
gs <- list(Bcells = c("MS4A1", "CD79B", "CD79A", "IGH1", "IGH2"),
           Tcells = c("CD3E", "CD3D", "CD3G", "CD7","CD8A"))

pbmc <- SeuratObject::pbmc_small |>
  runEscape(gene.sets = gs,
            min.size = NULL)

geyserEnrichment(pbmc,
                 assay = "escape",
                 gene.set = "Tcells")
```

`gseaEnrichment` *Classical GSEA-style Running-Enrichment Plot*

Description

Produces the familiar two-panel GSEA graphic—running enrichment score (RES) plus a "hit" rug—for a **single gene-set** evaluated across multiple biological groups (clusters, conditions, samples, ...).

Usage

```
gseaEnrichment(
  input.data,
  gene.set.use,
  gene.sets,
  group.by = NULL,
  summary.fun = "mean",
  p = 1,
  nperm = 1000,
  rug.height = 0.02,
  digits = 2,
  BPPARAM = NULL,
  palette = "inferno"
)
```

Arguments

`input.data`	A Seurat object or a SingleCellExperiment.
`gene.set.use`	Character(1). Name of the gene set to display.
`gene.sets`	A named list of character vectors, the result of [getGeneSets()], or the built-in data object [escape.gene.sets].
`group.by`	Metadata column. Defaults to the Seurat/SCE 'ident' slot when 'NULL'.
`summary.fun`	Method used to collapse expression within each group **before** ranking: one of '"mean"' (default), '"median"', '"max"', '"sum"', or '"geometric"'.
`p`	Weighting exponent in the KS statistic (classical GSEA uses 'p = 1').
`nperm`	Integer >= 0. Gene-label permutations per group (default 1000). '0' value will skip NES/*p* calculation.

rug.height	Vertical spacing of the hit rug as a fraction of the y-axis (default '0.02').
digits	Number of decimal places displayed for ES in the legend (default '2').
BPPARAM	A **BiocParallel** parameter object describing the parallel backend.
palette	Character. Any palette from hcl.pals.

Details

Algorithm (Subramanian _et al._, PNAS 2005) 1. Within every group, library-size-normalise counts to CPM. 2. Collapse gene expression with 'summary.fun' (mean/median/...). 3. Rank genes (descending) to obtain one ordered list per group. 4. Compute the weighted Kolmogorov–Smirnov running score (weight = \lstat\l^*p*). 5. ES = maximum signed deviation of the curve.

Value

A single 'patchwork'/'ggplot2' object

See Also

escape.matrix, densityEnrichment

Examples

```
pbmc_small <- SeuratObject::pbmc_small

gs <- list(Bcells = c("MS4A1", "CD79B", "CD79A", "IGH1", "IGH2"),
           Tcells = c("CD3E", "CD3D", "CD3G", "CD7","CD8A"))

gseaEnrichment(pbmc_small,
               gene.set.use = "Bcells",
               gene.sets    = gs,
               group.by     = "groups",
               summary.fun  = "mean",
               digits       = 3)
```

---

heatmapEnrichment *Visualize Enrichment Value Summaries Using Heatmaps*

Description

This function allows to the user to examine the heatmap with the mean enrichment values by group. The heatmap will have the gene sets as rows and columns will be the grouping variable.

Usage

```
heatmapEnrichment(
  input.data,
  assay = NULL,
  group.by = NULL,
  gene.set.use = "all",
  cluster.rows = FALSE,
  cluster.columns = FALSE,
```

```
  facet.by = NULL,
  scale = FALSE,
  summary.stat = "mean",
  palette = "inferno"
)
```

Arguments

input.data	Output of escape.matrix or a single-cell object previously processed by runEscape.
assay	Name of the assay holding enrichment scores when 'input.data' is a single-cell object. Ignored otherwise.
group.by	Metadata column plotted on the *x*-axis. Defaults to the Seurat/SCE 'ident' slot when 'NULL'.
gene.set.use	Vector of gene-set names to plot, or "all" (default) to show every available gene set.
cluster.rows, cluster.columns	Logical; if TRUE, rows/columns are ordered by Ward-linkage hierarchical clustering (Euclidean distance).
facet.by	Optional metadata column used to facet the plot.
scale	If TRUE, Z-transforms each gene-set column **after** summarization.
summary.stat	Method used to summarize expression within each
palette	Character. Any palette from hcl.pals.

Value

A ggplot2 object.

Examples

```
gs <- list(Bcells = c("MS4A1", "CD79B", "CD79A", "IGH1", "IGH2"),
           Tcells = c("CD3E", "CD3D", "CD3G", "CD7","CD8A"))

pbmc <- SeuratObject::pbmc_small |>
  runEscape(gene.sets = gs, min.size = NULL)

heatmapEnrichment(pbmc, assay = "escape", palette = "viridis")
```

---

pcaEnrichment *Visualize the PCA of Enrichment Values*

Description

This function allows to the user to examine the distribution of principal components run on the enrichment values.

Usage

```
pcaEnrichment(
  input.data,
  dimRed = NULL,
  x.axis = "PC1",
  y.axis = "PC2",
  facet.by = NULL,
  style = c("point", "hex"),
  add.percent.contribution = TRUE,
  display.factors = FALSE,
  number.of.factors = 10,
  palette = "inferno"
)
```

Arguments

`input.data`	Single-cell object (Seurat / SCE) **or** the raw list returned by [‘performPCA()‘].
`dimRed`	Name of the dimensional-reduction slot to pull from a single-cell object. Ignored when ‘input.data‘ is the list output.
`x.axis, y.axis`	Character vectors naming the PCs to display (e.g. "PC1").
`facet.by`	Metadata column to facet plot.
`style`	"point" (default) or "hex".
`add.percent.contribution`	Include percent variance explained in axis labels.
`display.factors`	Draw arrows for the top gene-set loadings.
`number.of.factors`	Integer; how many loadings to display if ‘display.factors = TRUE‘.
`palette`	Character. Any palette from `hcl.pals`.
	#’ @examples GS <- list(Bcells = c("MS4A1", "CD79B", "CD79A", "IGH1", "IGH2"), Tcells = c("CD3E", "CD3D", "CD3G", "CD7","CD8A")) pbmc_small <- SeuratObject::pbmc_small pbmc_small <- runEscape(pbmc_small, gene.sets = GS, min.size = NULL)
	pbmc_small <- performPCA(pbmc_small, assay = "escape")
	pcaEnrichment(pbmc_small, x.axis = "PC1", y.axis = "PC2", dimRed = "escape.PCA")

Value

A **ggplot2** object.

`performNormalization` *Perform Normalization on Enrichment Data*

Description

Scales each enrichment value by the **number of genes from the set that are expressed** in that cell (non-zero counts). Optionally shifts results into a positive range and/or applies a natural-log transform for compatibility with log-based differential tests.

Usage

```
performNormalization(
  input.data,
  enrichment.data = NULL,
  assay = "escape",
  gene.sets = NULL,
  make.positive = FALSE,
  scale.factor = NULL,
  groups = NULL
)
```

Arguments

`input.data`	raw-counts matrix (‘genes × cells‘), a Seurat object, or a SingleCellExperiment. Gene identifiers must match those in ‘gene.sets‘.
`enrichment.data`	Output of `escape.matrix` or a single-cell object previously processed by `runEscape`.
`assay`	Name of the assay holding enrichment scores when ‘input.data‘ is a single-cell object. Ignored otherwise.
`gene.sets`	A named list of character vectors, the result of [getGeneSets()], or the built-in data object [escape.gene.sets]. List names become column names in the result.
`make.positive`	Logical; if ‘TRUE‘ shifts each column so its minimum is zero.
`scale.factor`	Optional numeric vector overriding gene-count scaling (length = #cells). Use when you want external per-cell normalization factors.
`groups`	Integer >= 1. Number of cells per processing chunk. Larger values reduce overhead but increase memory usage. Default **1000**.

Value

If ‘input.data‘ is an object, the same object with a new assay "<assay>_normalized". Otherwise a matrix of normalized scores.

Examples

```
gs <- list(Bcells = c("MS4A1", "CD79B", "CD79A", "IGH1", "IGH2"),
           Tcells = c("CD3E", "CD3D", "CD3G", "CD7","CD8A"))

pbmc <- SeuratObject::pbmc_small |>
  runEscape(gene.sets = gs,
            min.size = NULL)
```

```
pbmc <- performNormalization(pbmc,
                             assay = "escape",
                             gene.sets = gs)
```

`performPCA` *Perform Principal Component Analysis on Enrichment Data*

Description

This function allows users to calculate the principal components for the gene set enrichment values. For single-cell data, the PCA will be stored with the dimensional reductions. If a matrix is used as input, the output is a list for further plotting. Alternatively, users can use functions for PCA calculations based on their desired workflow in lieu of using `performPCA`, but will not be compatible with downstream `pcaEnrichment` visualization.

Usage

```
performPCA(
  input.data,
  assay = "escape",
  scale = TRUE,
  n.dim = 10,
  reduction.name = "escape.PCA",
  reduction.key = "escPC_"
)
```

Arguments

`input.data`	Output of `escape.matrix` or a single-cell object previously processed by `runEscape`.
`assay`	Name of the assay holding enrichment scores when ‘input.data‘ is a single-cell object. Ignored otherwise.
`scale`	Logical; if ‘TRUE‘ standardises each gene-set column before PCA.
`n.dim`	Integer >= 1 or vector; the **largest** value sets the number of principal components to compute / keep.
`reduction.name, reduction.key`	Names used when writing back to a Seurat / SCE object.

Value

If ‘input.data‘ is a single-cell object, the same object with a new dimensional-reduction slot. *Otherwise* a list with ‘PCA‘, ‘eigen_values‘, ‘contribution‘, and ‘rotation‘.

Examples

```
gs <- list(Bcells = c("MS4A1", "CD79B", "CD79A", "IGH1", "IGH2"),
           Tcells = c("CD3E", "CD3D", "CD3G", "CD7","CD8A"))

pbmc <- SeuratObject::pbmc_small |>
  runEscape(gene.sets = gs,
```

```
            min.size = NULL)

pbmc <- performPCA(pbmc,
                   assay = "escape")
```

`ridgeEnrichment` *Visualize Enrichment Distributions Using Ridge Plots*

Description

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

Usage

```
ridgeEnrichment(
  input.data,
  gene.set.use,
  assay = NULL,
  group.by = NULL,
  color.by = "group",
  order.by = NULL,
  scale = FALSE,
  facet.by = NULL,
  add.rug = FALSE,
  palette = "inferno"
)
```

Arguments

`input.data`	Output of `escape.matrix` or a single-cell object previously processed by `runEscape`.
`gene.set.use`	Character(1). Name of the gene set to display.
`assay`	Name of the assay holding enrichment scores when ‘input.data‘ is a single-cell object. Ignored otherwise.
`group.by`	Metadata column plotted on the *y*-axis. Defaults to the Seurat/SCE ‘ident‘ slot when ‘NULL‘.
`color.by`	Aesthetic mapped to point color. Use either *"group"* (default = ‘group.by‘) for categorical coloring or the *name of a gene-set* (e.g. same as ‘gene.set‘) to obtain a numeric accepted.
`order.by`	How to arrange the x-axis: *‘"mean"‘* – groups ordered by decreasing group mean; *‘"group"‘* – natural sort of group labels; *‘NULL‘* – keep original ordering.
`scale`	Logical; if ‘TRUE‘ scores are centred/scaled (Z-score) prior to plotting.
`facet.by`	Optional metadata column used to facet the plot.
`add.rug`	Logical. Draw per-cell tick marks underneath each ridge.
`palette`	Character. Any palette from `hcl.pals`.

Value

A [ggplot2] object.

Examples

```
gs <- list(Bcells = c("MS4A1", "CD79B", "CD79A", "IGH1", "IGH2"),
           Tcells = c("CD3E", "CD3D", "CD3G", "CD7","CD8A"))

pbmc <- SeuratObject::pbmc_small |>
  runEscape(gene.sets = gs, min.size = NULL)

ridgeEnrichment(pbmc, assay = "escape",
                gene.set.use = "Tcells",
                group.by = "groups")
```

`runEscape` *Calculate Enrichment Scores Using Seurat or SingleCellExperiment Objects*

Description

‘runEscape()‘ is a convenience wrapper around [escape.matrix()] that computes enrichment scores and inserts them as a new assay (default ‘"escape"‘) in a **Seurat** or **SingleCellExperiment** object. All arguments (except ‘new.assay.name‘) map directly to their counterparts in ‘escape.matrix()‘.

Usage

```
runEscape(
  input.data,
  gene.sets,
  method = c("ssGSEA", "GSVA", "UCell", "AUCell"),
  groups = 1000,
  min.size = 5,
  normalize = FALSE,
  make.positive = FALSE,
  new.assay.name = "escape",
  min.expr.cells = 0,
  min.filter.by = NULL,
  BPPARAM = NULL,
  ...
)
```

Arguments

`input.data`	A raw-counts matrix (‘genes × cells‘), a Seurat object, or a SingleCellExperiment. Gene identifiers must match those in ‘gene.sets‘.
`gene.sets`	A named list of character vectors, the result of [getGeneSets()], or the built-in data object [escape.gene.sets]. List names become column names in the result.
`method`	Scoring algorithm (case-insensitive). One of ‘"GSVA"‘, ‘"ssGSEA"‘, ‘"UCell"‘, or ‘"AUCell"‘. Default **‘"ssGSEA"‘**.

groups	Integer >= 1. Number of cells per processing chunk. Larger values reduce overhead but increase memory usage. Default **1000**.
min.size	Minimum number of genes from a set that must be detected in the expression matrix for that set to be scored. Default **5**. Use 'NULL' to disable filtering.
normalize	Logical. If 'TRUE', the score matrix is passed to [performNormalization()] (drop-out scaling and optional log transform). Default **FALSE**.
make.positive	Logical. If 'TRUE' *and* 'normalize = TRUE', shifts every gene-set column so its global minimum is zero, facilitating downstream log-ratio analyses. Default **FALSE**.
new.assay.name	Character. Name for the assay that will store the enrichment matrix in the returned object. Default **"escape"**.
min.expr.cells	Numeric. Gene-expression filter threshold (see details above). Default **0** (no gene filtering).
min.filter.by	Character or 'NULL'. Column name in 'meta.data' (Seurat) or 'colData' (SCE) defining groups within which the 'min.expr.cells' rule is applied. Default **'NULL'**.
BPPARAM	A **BiocParallel** parameter object describing the parallel backend.
...	Extra arguments passed verbatim to the chosen back-end scoring function ('gsva()', 'ScoreSignatures_UCell()', or 'AUCell_calcAUC()').

Value

The input single-cell object with an additional assay containing the enrichment scores ('cells × gene-sets'). Matrix orientation follows standard single-cell conventions (gene-sets as rows inside the assay).

Author(s)

Nick Borcherding, Jared Andrews

See Also

[escape.matrix()] for the underlying computation, [performNormalization()] to add normalized scores, [heatmapEnrichment()], [ridgeEnrichment()] and related plotting helpers for visualization.

Examples

```
gs <- list(Bcells = c("MS4A1", "CD79B", "CD79A", "IGH1", "IGH2"),
           Tcells = c("CD3E", "CD3D", "CD3G", "CD7","CD8A"))

sce <- SeuratObject::pbmc_small
sce <- runEscape(sce,
                 gene.sets = gs,
                 method = "GSVA",
                 groups = 1000,
                 min.size = 3,
                 new.assay.name = "escape")
```

`scatterEnrichment` *Plot 2D Enrichment Distributions With Density or Hexplots*

Description

Visualize the relationship between *two* enrichment scores at single-cell resolution. By default points are shaded by local 2-D density (‘color.by = "density"‘), but users can instead color by a metadata column (discrete) or by the raw gene-set scores themselves (continuous).

Usage

```
scatterEnrichment(
  input.data,
  assay = NULL,
  x.axis,
  y.axis,
  facet.by = NULL,
  group.by = NULL,
  color.by = c("density", "group", "x", "y"),
  style = c("point", "hex"),
  scale = FALSE,
  bins = 40,
  point.size = 1.2,
  alpha = 0.8,
  palette = "inferno",
  add.corr = FALSE
)
```

Arguments

`input.data`	Output of `escape.matrix` or a single-cell object previously processed by `runEscape`.
`assay`	Name of the assay holding enrichment scores when ‘input.data‘ is a single-cell object. Ignored otherwise.
`x.axis, y.axis`	Gene-set names to plot on the *x* and *y* axes.
`facet.by`	Optional metadata column used to facet the plot.
`group.by`	Metadata column plotted. Defaults to the Seurat/SCE ‘ident‘ slot when ‘NULL‘.
`color.by`	Aesthetic mapped to point color. Use ‘"density"‘ (default), ‘"group"‘, ‘"x"‘, or ‘"y"‘. The latter two apply a continuous gradient to the corresponding axis.
`style`	‘"point"‘ (density-aware points) or ‘"hex"‘ (hex-bin).
`scale`	Logical; if ‘TRUE‘ scores are centered/scaled (Z-score) prior to plotting.
`bins`	Number of hex bins along each axis when ‘style = "hex"‘.
`point.size, alpha`	Aesthetic tweaks for ‘style = "point"‘.
`palette`	Character. Any palette from `hcl.pals`.
`add.corr`	Logical. Add Pearson and Spearman correlation coefficients (top-left corner of the first facet).

Value

A **ggplot2** object.

Examples

```
gs <- list(
  Bcells = c("MS4A1","CD79B","CD79A","IGH1","IGH2"),
  Tcells = c("CD3E","CD3D","CD3G","CD7","CD8A")
)
pbmc <- SeuratObject::pbmc_small |>
  runEscape(gene.sets = gs, min.size = NULL)

scatterEnrichment(
  pbmc,
  assay     = "escape",
  x.axis    = "Tcells",
  y.axis    = "Bcells",
  color.by  = "group",
  group.by  = "groups",
  add.corr  = TRUE,
  point.size = 1
)
```

---

splitEnrichment *Plot Enrichment Distributions Using Split or Dodged Violin Plots*

Description

Visualize the distribution of gene set enrichment scores across groups using violin plots. When ‘split.by‘ contains exactly two levels, the function draws split violins for easy group comparison within each ‘group.by‘ category. If ‘split.by‘ has more than two levels, standard dodged violins are drawn instead.

Usage

```
splitEnrichment(
  input.data,
  assay = NULL,
  split.by = NULL,
  group.by = NULL,
  gene.set.use = NULL,
  order.by = NULL,
  facet.by = NULL,
  scale = TRUE,
  palette = "inferno"
)
```

Arguments

input.data	Output of escape.matrix or a single-cell object previously processed by runEscape.

assay	Name of the assay holding enrichment scores when ‘input.data‘ is a single-cell object. Ignored otherwise.
split.by	A metadata column used to split or color violins. Must contain at least two levels. If it contains more than two, dodged violins are used.
group.by	Metadata column plotted on the *x*-axis. Defaults to the Seurat/SCE ‘ident‘ slot when ‘NULL‘.
gene.set.use	Character(1). Name of the gene set to display.
order.by	How to arrange the x-axis: *‘"mean"‘* – groups ordered by decreasing group mean; *‘"group"‘* – natural sort of group labels; *‘NULL‘* – keep original ordering.
facet.by	Optional metadata column used to facet the plot.
scale	Logical; if ‘TRUE‘ scores are centred/scaled (Z-score) prior to plotting.
palette	Character. Any palette from hcl.pals.

Value

A [ggplot2] object.

Examples

```
gs <- list(Bcells = c("MS4A1", "CD79B", "CD79A", "IGH1", "IGH2"),
           Tcells = c("CD3E", "CD3D", "CD3G", "CD7","CD8A"))

pbmc <- SeuratObject::pbmc_small |>
  runEscape(gene.sets = gs, min.size = NULL)

splitEnrichment(input.data = pbmc,
                assay = "escape",
                split.by = "groups",
                gene.set.use = "Tcells")
```

Index

∗ **datasets**
escape.gene.sets, 5

densityEnrichment, 2, *11*

enrichIt, 3
enrichItPlot, 4
escape.gene.sets, 5
escape.matrix, 6, *9*, *11, 12*, *14–16*, *19, 20*

getGeneSets, 8
geyserEnrichment, 9
gseaEnrichment, 10

hcl.pals, *2*, *5*, *9*, *11–13*, *16*, *19*, *21*
heatmapEnrichment, 11

pcaEnrichment, 12, *15*
performNormalization, 14
performPCA, *15*, 15

ridgeEnrichment, 16
runEscape, *9*, *12*, *14–16*, 17, *19, 20*

scatterEnrichment, 19
Seurat, *2*, *6*, *10*, *14*, *17*
SingleCellExperiment, *2*, *6*, *10*, *14*, *17*
splitEnrichment, 20