Package: ASURAT (via r-universe)

June 9, 2026

Type Package

Title Functional annotation-driven unsupervised clustering for single-cell data

Version 1.16.0

Description ASURAT is a software for single-cell data analysis. Using ASURAT, one can simultaneously perform unsupervised clustering and biological interpretation in terms of cell type, disease, biological process, and signaling pathway activity. Inputting a single-cell RNA-seq data and knowledge-based databases, such as Cell Ontology, Gene Ontology, KEGG, etc., ASURAT transforms gene expression tables into original multivariate tables, termed sign-by-sample matrices (SSMs).

License GPL-3 + file LICENSE

biocViews GeneExpression, SingleCell, Sequencing, Clustering, GeneSignaling

VignetteBuilder knitr

Encoding UTF-8

LazyData TRUE

Depends R (>= 4.0.0)

Imports SingleCellExperiment, SummarizedExperiment, S4Vectors, Rcpp (>= 1.0.7), cluster, utils, plot3D, ComplexHeatmap, circlize, grid, grDevices, graphics

Suggests ggplot2, TENxPBMCData, dplyr, Rtsne, Seurat, AnnotationDbi, BiocGenerics, stringr, org.Hs.eg.db, knitr, rmarkdown, testthat (>= 3.0.0)

RoxygenNote 7.1.2

LinkingTo Rcpp

Config/testthat/edition 3

Config/pak/sysreqs libpng-dev perl zlib1g-dev

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

Date/Publication 2026-04-28 12:58:02 UTC

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

RemoteRef RELEASE_3_23

RemoteSha c4b67bbfb3159f51286cfe29e252a9fc7b34f1e9

Contents

`add_metadata` *Add metadata of variables and samples.*

Description

This function adds metadata of variables and samples.

Usage

```
add_metadata(sce = NULL, mitochondria_symbol = NULL)
```

Arguments

sce	A SingleCellExperiment object.
mitochondria_symbol	A string representing for mitochondrial genes. This function computes percents of reads that map to the mitochondrial genes. Examples are '^MT-', '^mt-', etc.

Value

A SingleCellExperiment object.

Examples

```
data(pbmc_eg)
pbmc <- add_metadata(sce = pbmc_eg, mitochondria_symbol = "^MT-")
```

ASURAT *Functional annotation-driven unsupervised clustering of SingleCell data.*

Description

ASURAT is a software for single-cell data analysis. Using ASURAT, one can simultaneously perform unsupervised clustering and biological interpretation in terms of cell type, disease, biological process, and signaling pathway activity. Inputting a single-cell RNA-seq data and knowledge-based databases, such as Cell Ontology, Gene Ontology, KEGG, etc., ASURAT transforms gene expression tables into original multivariate tables, termed sign-by-sample matrices (SSMs).

---

bubble_sort *Perform bubble sorting, counting the number of steps.*

Description

Perform bubble sorting, counting the number of steps.

Usage

```
bubble_sort(listdata)
```

Arguments

listdata	A list of vector and integer. For example, in R code, listdata = list(vec = c(1, 0, 1, ...), cnt = 0). The integer (cnt = 0) is the initial number of steps for bubble sorting.

Value

A List.

Examples

```
bubble_sort(list(vec = c(1, 1, 0), cnt = 0))
```

`cluster_genesets` *Cluster each functional gene set into three groups.*

Description

This function clusters each functional gene set into strongly, variably, and weakly correlated gene sets.

Usage

```
cluster_genesets(sce = NULL, cormat = NULL, th_posi = NULL, th_nega = NULL)
```

Arguments

`sce`	A SingleCellExperiment object.
`cormat`	A correlation matrix of gene expressions.
`th_posi`	A threshold of positive correlation coefficient.
`th_nega`	A threshold of negative correlation coefficient.

Value

A SingleCellExperiment object.

Examples

```
data(pbmc_eg)
data(human_GO_eg)
mat <- t(as.matrix(SummarizedExperiment::assay(pbmc_eg, "centered")))
pbmc_cormat <- cor(mat, method = "spearman")
pbmcs <- list(GO = pbmc_eg)
S4Vectors::metadata(pbmcs$GO) <- list(sign = human_GO_eg[["BP"]])
pbmcs$GO <- remove_signs(sce = pbmcs$GO, min_ngenes = 2, max_ngenes = 1000)
pbmcs$GO <- cluster_genesets(sce = pbmcs$GO, cormat = pbmc_cormat,
                             th_posi = 0.24, th_nega = -0.20)
# The results are stored in `metadata(pbmcs$GO)$sign`.
```

compute_sepI_all *Compute separation indices for each cluster against the others.*

Description

This function computes separation indices for each cluster versus the others.

Usage

```
compute_sepI_all(sce = NULL, labels = NULL, nrand_samples = NULL)
```

Arguments

sce	A SingleCellExperiment object.
labels	A vector of labels of all the samples (cells).
nrand_samples	An integer for the number of samples used for random sampling, which samples at least one sample per cluster.

Value

A SingleCellExperiment object.

Examples

```
data(pbmcs_eg)
labels <- SummarizedExperiment::colData(pbmcs_eg$GO)$seurat_clusters
pbmcs_eg$GO <- compute_sepI_all(sce = pbmcs_eg$GO, labels = labels,
                                nrand_samples = 10)
# The results are stored in `metadata(pbmcs_eg$GO)$marker_signs`.
```

compute_sepI_clusters *Compute separation indices of sign scores for given two clusters.*

Description

This function computes separation indices of sign scores for given two clusters.

Usage

```
compute_sepI_clusters(
  sce = NULL,
  labels = NULL,
  nrand_samples = NULL,
  ident_1 = NULL,
  ident_2 = NULL
)
```

Arguments

sce	A SingleCellExperiment object.
labels	A vector of labels of all the samples.
nrand_samples	An integer for the number of samples used for random sampling, which samples at least one sample per cluster.
ident_1	Label names identifying cluster numbers, e.g., ident_1 = 1, ident_1 = c(1, 3).
ident_2	Label names identifying cluster numbers, e.g., ident_2 = 2, ident_2 = c(2, 4).

Value

A SingleCellExperiment object.

Examples

```
data(pbmcs_eg)
labels <- SummarizedExperiment::colData(pbmcs_eg$GO)$seurat_clusters
pbmcs_eg$GO <- compute_sepI_clusters(sce = pbmcs_eg$GO, labels = labels,
                                     nrand_samples = 10, ident_1 = 1,
                                     ident_2 = c(0, 2))
# The results are stored in `metadata(pbmcs_eg$GO)$marker_signs`.
```

---

create_signs *Define signs for strongly and variably correlated gene sets.*

Description

This function define signs for strongly and variably correlated gene sets.

Usage

```
create_signs(sce = NULL, min_cnt_strg = 2, min_cnt_vari = 2)
```

Arguments

sce	A SingleCellExperiment object.
min_cnt_strg	An integer for the cutoff value for strongly correlated gene sets.
min_cnt_vari	An integer for the cutoff value for variably correlated gene sets.

Value

A SingleCellExperiment object.

Examples

```
data(pbmc_eg)
data(human_GO_eg)
mat <- t(as.matrix(SummarizedExperiment::assay(pbmc_eg, "centered")))
pbmc_cormat <- cor(mat, method = "spearman")
pbmcs <- list(GO = pbmc_eg)
S4Vectors::metadata(pbmcs$GO) <- list(sign = human_GO_eg[["BP"]])
pbmcs$GO <- remove_signs(sce = pbmcs$GO, min_ngenes = 2, max_ngenes = 1000)
pbmcs$GO <- cluster_genesets(sce = pbmcs$GO, cormat = pbmc_cormat,
                             th_posi = 0.24, th_nega = -0.20)
pbmcs$GO <- create_signs(sce = pbmcs$GO, min_cnt_strg = 2, min_cnt_vari = 2)
# The results are stored in `metadata(pbmcs$GO)$sign_all`.
```

---

human_COMSig_eg *A list of small Cell Ontology and MSigDB databases for human.*

Description

A list of small Cell Ontology and MSigDB databases for human.

Usage

```
human_COMSig_eg
```

Format

A list of dataframe.

---

human_GO_eg *A list of small Gene Ontology database for human.*

Description

A list of small Gene Ontology database for human.

Usage

```
human_GO_eg
```

Format

A list of dataframe.

human_KEGG_eg *A list of small KEGG database for human.*

Description

A list of small KEGG database for human.

Usage

```
human_KEGG_eg
```

Format

A list of dataframe.

makeSignMatrix *Create a new SingleCellExperiment object for sign-by-sample matrices.*

Description

This function creates a new SingleCellExperiment object for sign-by-sample matrices (SSM) by concatenating SSMs for strongly and variably correlated gene sets.

Usage

```
makeSignMatrix(sce = NULL, weight_strg = 0.5, weight_vari = 0.5)
```

Arguments

sce	A SingleCellExperiment object.
weight_strg	A weight parameter for strongly correlated gene sets.
weight_vari	A weight parameter for variably correlated gene sets.

Value

A SingleCellExperiment object.

Examples

```
data(pbmc_eg)
data(human_GO_eg)
mat <- t(as.matrix(SummarizedExperiment::assay(pbmc_eg, "centered")))
pbmc_cormat <- cor(mat, method = "spearman")
pbmcs <- list(GO = pbmc_eg)
S4Vectors::metadata(pbmcs$GO) <- list(sign = human_GO_eg[["BP"]])
pbmcs$GO <- remove_signs(sce = pbmcs$GO, min_ngenes = 2, max_ngenes = 1000)
pbmcs$GO <- cluster_genesets(sce = pbmcs$GO, cormat = pbmc_cormat,
                             th_posi = 0.24, th_nega = -0.20)
pbmcs$GO <- create_signs(sce = pbmcs$GO, min_cnt_strg = 2, min_cnt_vari = 2)
pbmcs$GO <- makeSignMatrix(sce = pbmcs$GO, weight_strg = 0.5,
                           weight_vari = 0.5)
# The resutls can be check by, e.g., assay(pbmcs$GO, "counts").
```

---

pbmc_eg *A SingleCellExperiment object made from a gene expression table.*

Description

A SingleCellExperiment object, including 50 genes and 50 cells. The original data "4k PBMCs from a Healthy Donor" was downloaded from 10x Genomics database.

Usage

```
pbmc_eg
```

Format

SingleCellExperiment object.

Source

https://support.10xgenomics.com/single-cell-gene-expression

---

pbmcs_eg *A list of SingleCellExperiment objects made from sign-sample matrices.*

Description

A list of SingleCellExperiment objects, consisting of small sign-by-sample matrices, pbmcs_eg$CM (using Cell Ontology and MSigDB databases), pbmcs_eg$GO (using Gene Ontology database), and pbmcs_eg$KG (KEGG). Here, pbmcs_eg$CM, pbmcs_eg$GO, and pbmcs_eg$KG include 87, 72, and 64 signs, respectively, and 50 cells.

Usage

```
pbmcs_eg
```

Format

A list of SingleCellExperiment objects.

`plot_dataframe3D` *Visualize a three-dimensional data with labels and colors.*

Description

This function visualizes a three-dimensional data with labels and colors.

Usage

```
plot_dataframe3D(
  dataframe3D = NULL,
  labels = NULL,
  colors = NULL,
  theta = 30,
  phi = 30,
  title = "",
  xlabel = "",
  ylabel = "",
  zlabel = ""
)
```

Arguments

`dataframe3D`	A dataframe with three columns.
`labels`	NULL or a vector of labels of all the samples, corresponding to colors.
`colors`	NULL or a vector of colors of all the samples, corresponding to labels.
`theta`	Angle of the plot.
`phi`	Angle of the plot.
`title`	Title.
`xlabel`	x-axis label.
`ylabel`	y-axis label.
`zlabel`	z-axis label.

Value

A scatter3D object in plot3D package.

Examples

```
data(pbmcs_eg)
mat <- SingleCellExperiment::reducedDim(pbmcs_eg$CM, "UMAP")[, 1:3]
dataframe3D <- as.data.frame(mat)
labels <- SummarizedExperiment::colData(pbmcs_eg$CM)$seurat_clusters
plot_dataframe3D(dataframe3D = dataframe3D, labels = labels, colors = NULL,
                 theta = 45, phi = 20, title = "PBMC (CO & MSigDB)",
                 xlabel = "UMAP_1", ylabel = "UMAP_2", zlabel = "UMAP_3")
```

---

plot_multiheatmaps *Visualize multivariate data by heatmaps.*

Description

This function visualizes multivariate data by heatmaps.

Usage

```
plot_multiheatmaps(
  ssm_list = NULL,
  gem_list = NULL,
  ssmlabel_list = NULL,
  gemlabel_list = NULL,
  nrand_samples = NULL,
  show_row_names = FALSE,
  title = NULL
)
```

Arguments

ssm_list	A list of sign-by-sample matrices.
gem_list	A list of gene-by-sample matrices.
ssmlabel_list	NULL or a list of dataframes of sample (cell) labels and colors. The length of the list must be as same as that of ssm_list, and the order of labels in each list must be as same as those in ssm_list.
gemlabel_list	NULL or a list of dataframes of sample (cell) annotations and colors. The length of the list must be as same as that of gem_list, and the order of labels in each list must be as same as those in gem_list.
nrand_samples	Number of samples (cells) used for random sampling.
show_row_names	TRUE or FALSE: if TRUE, row names are shown.
title	Title.

Value

A ComplexHeatmap object.

Examples

```
data(pbmcs_eg)
mat_CM <- SummarizedExperiment::assay(pbmcs_eg$CM, "counts")
mat_GO <- SummarizedExperiment::assay(pbmcs_eg$GO, "counts")
mat_KG <- SummarizedExperiment::assay(pbmcs_eg$KG, "counts")
ssm_list <- list(SSM_COMSig = mat_CM, SSM_GO = mat_GO, SSM_KEGG = mat_KG)
se <- SingleCellExperiment::altExp(pbmcs_eg$CM, "logcounts")
mat <- SummarizedExperiment::assay(se, "counts")
se <- SingleCellExperiment::altExp(pbmcs_eg$CM, "logcounts")
gem_list <- list(GeneExpr = SummarizedExperiment::assay(se, "counts"))
labels <- list() ; ssmlabel_list <- list()
for(i in seq_along(pbmcs_eg)){
  fa <- SummarizedExperiment::colData(pbmcs_eg[[i]])$seurat_clusters
  labels[[i]] <- data.frame(label = fa)
  colors <- rainbow(length(unique(labels[[i]]$label)))[labels[[i]]$label]
  labels[[i]]$color <- colors
  ssmlabel_list[[i]] <- labels[[i]]
}
names(ssmlabel_list) <- c("Label_COMSig", "Label_GO", "Label_KEGG")
phases <- SummarizedExperiment::colData(pbmcs_eg$CM)$Phase
label_CC <- data.frame(label = phases, color = NA)
gemlabel_list <- list(CellCycle = label_CC)
plot_multiheatmaps(ssm_list = ssm_list, gem_list = gem_list,
                   ssmlabel_list = ssmlabel_list,
                   gemlabel_list = gemlabel_list, nrand_samples = 50,
                   show_row_names = FALSE, title = "PBMC")
```

---

remove_samples *Remove samples based on expression profiles across variables.*

Description

This function removes sample data by setting minimum and maximum threshold values for the metadata.

Usage

```
remove_samples(
  sce = NULL,
  min_nReads = NULL,
  max_nReads = NULL,
  min_nGenes = NULL,
  max_nGenes = NULL,
  min_percMT = NULL,
  max_percMT = NULL
)
```

Arguments

sce	A SingleCellExperiment object.
min_nReads	A minimum threshold value of the number of reads.
max_nReads	A maximum threshold value of the number of reads.
min_nGenes	A minimum threshold value of the number of non-zero expressed genes.
max_nGenes	A maximum threshold value of the number of non-zero expressed genes.
min_percMT	A minimum threshold value of the percent of reads that map to mitochondrial genes.
max_percMT	A maximum threshold value of the percent of reads that map to mitochondrial genes.

Value

A SingleCellExperiment object.

Examples

```
data(pbmc_eg)
pbmc <- add_metadata(sce = pbmc_eg, mitochondria_symbol = "^MT-")
pbmc <- remove_samples(sce = pbmc, min_nReads = 0, max_nReads = 1e+10,
                       min_nGenes = 0, max_nGenes = 1e+10,
                       min_percMT = NULL, max_percMT = NULL)
```

---

remove_signs *Remove signs including too few or too many genes.*

Description

This function removes signs including too few or too many genes.

Usage

```
remove_signs(sce = NULL, min_ngenes = 2, max_ngenes = 1000)
```

Arguments

sce	A SingleCellExperiment object.
min_ngenes	Minimum number of genes, which must be greater than one.
max_ngenes	Maximum number of genes, which must be greater than one.

Value

A SingleCellExperiment object.

Examples

```
data(pbmc_eg)
data(human_GO_eg)
pbmcs <- list(GO = pbmc_eg)
S4Vectors::metadata(pbmcs$GO) <- list(sign = human_GO_eg[["BP"]])
pbmcs$GO <- remove_signs(sce = pbmcs$GO, min_ngenes = 2, max_ngenes = 1000)
# The results are stored in `metadata(pbmcs$GO)$sign`.
```

`remove_signs_manually` *Remove signs by specifying keywords.*

Description

This function removes signs by specifying keywords.

Usage

```
remove_signs_manually(sce = NULL, keywords = NULL)
```

Arguments

`sce`	A SingleCellExperiment object.
`keywords`	keywords separated by pipes '|'.

Value

A SingleCellExperiment object.

Examples

```
data(pbmc_eg)
data(human_GO_eg)
mat <- t(as.matrix(SummarizedExperiment::assay(pbmc_eg, "centered")))
pbmc_cormat <- cor(mat, method = "spearman")
pbmcs <- list(GO = pbmc_eg)
S4Vectors::metadata(pbmcs$GO) <- list(sign = human_GO_eg[["BP"]])
pbmcs$GO <- remove_signs(sce = pbmcs$GO, min_ngenes = 2, max_ngenes = 1000)
pbmcs$GO <- cluster_genesets(sce = pbmcs$GO, cormat = pbmc_cormat,
                             th_posi = 0.24, th_nega = -0.20)
pbmcs$GO <- create_signs(sce = pbmcs$GO, min_cnt_strg = 2, min_cnt_vari = 2)
keywords <- "Covid19|foofoo|hogehoge"
pbmcs$GO <- remove_signs_manually(sce = pbmcs$GO, keywords = keywords)
# The results are stored in `metadata(pbmcs$GO)$sign_SCG`,
# `metadata(pbmcs$GO)$sign_VCG`, and `metadata(pbmcs$GO)$sign_all`.
```

`remove_signs_redundant` *Remove redundant signs using semantic similarity matrices.*

Description

This function removes redundant signs using semantic similarity matrices.

Usage

```
remove_signs_redundant(
  sce = NULL,
  similarity_matrix = NULL,
  threshold = NULL,
  keep_rareID = NULL
)
```

Arguments

`sce`	A SingleCellExperiment object.
`similarity_matrix`	A semantic similarity matrix.
`threshold`	A threshold value of semantic similarity, used for regarding biological terms as similar ones
`keep_rareID`	If TRUE, biological terms with the larger ICs are kept.

Value

A SingleCellExperiment object.

Examples

```
data(pbmc_eg)
data(human_GO_eg)
mat <- t(as.matrix(SummarizedExperiment::assay(pbmc_eg, "centered")))
pbmc_cormat <- cor(mat, method = "spearman")
pbmcs <- list(GO = pbmc_eg)
S4Vectors::metadata(pbmcs$GO) <- list(sign = human_GO_eg[["BP"]])
pbmcs$GO <- remove_signs(sce = pbmcs$GO, min_ngenes = 2, max_ngenes = 1000)
pbmcs$GO <- cluster_genesets(sce = pbmcs$GO, cormat = pbmc_cormat,
                             th_posi = 0.24, th_nega = -0.20)
pbmcs$GO <- create_signs(sce = pbmcs$GO, min_cnt_strg = 2, min_cnt_vari = 2)
pbmcs$GO <- remove_signs_redundant(
  sce = pbmcs$GO, similarity_matrix = human_GO_eg$similarity_matrix$BP,
  threshold = 0.80, keep_rareID = TRUE)
# The results are stored in `metadata(pbmcs$GO)$sign_SCG`,
# `metadata(pbmcs$GO)$sign_VCG`, `metadata(pbmcs$GO)$sign_all`,
# and if there exist, `metadata(pbmcs$GO)$sign_SCG_redundant` and
```

```
# `metadata(pbmcs$GO)$sign_VCG_redundant`.
```

`remove_variables` *Remove variables based on expression profiles across samples.*

Description

This function removes low expressed variable data.

Usage

```
remove_variables(sce = NULL, min_nsamples = 0)
```

Arguments

`sce`	A SingleCellExperiment object.
`min_nsamples`	An integer. This function removes variables for which the numbers of non-zero expressing samples are less than this value.

Value

A SingleCellExperiment object.

Examples

```
data(pbmc_eg)
pbmc <- add_metadata(sce = pbmc_eg, mitochondria_symbol = "^MT-")
pbmc <- remove_variables(sce = pbmc, min_nsamples = 10)
```

`remove_variables_second`
Remove variables based on the mean expression levels across samples.

Description

This function removes variable data such that the mean expression levels across samples are less than ‘min_meannReads‘.

Usage

```
remove_variables_second(sce = NULL, min_meannReads = 0)
```

Arguments

sce	A SingleCellExperiment object.
min_meannReads	An integer. This function removes variables for which the mean read counts are less than this value.

Value

A SingleCellExperiment object.

Examples

```
data(pbmc_eg)
pbmc <- remove_variables_second(sce = pbmc_eg, min_meannReads = 0.01)
```

---

select_signs_manually *Select signs by specifying keywords.*

Description

This function selects signs by specifying keywords.

Usage

```
select_signs_manually(sce = NULL, keywords = NULL)
```

Arguments

sce	An ASURAT object.
keywords	Keywords separated by a pipe.

Value

An ASURAT object.

Examples

```
data(pbmc_eg)
data(human_GO_eg)
mat <- t(as.matrix(SummarizedExperiment::assay(pbmc_eg, "centered")))
pbmc_cormat <- cor(mat, method = "spearman")
pbmcs <- list(GO = pbmc_eg)
S4Vectors::metadata(pbmcs$GO) <- list(sign = human_GO_eg[["BP"]])
pbmcs$GO <- remove_signs(sce = pbmcs$GO, min_ngenes = 2, max_ngenes = 1000)
pbmcs$GO <- cluster_genesets(sce = pbmcs$GO, cormat = pbmc_cormat,
                             th_posi = 0.24, th_nega = -0.20)
pbmcs$GO <- create_signs(sce = pbmcs$GO, min_cnt_strg = 2, min_cnt_vari = 2)
keywords <- "cell|process"
```

```
pbmcs$GO <- select_signs_manually(sce = pbmcs$GO, keywords = keywords)
# The results are stored in `metadata(pbmcs$GO)$sign_SCG`,
# `metadata(pbmcs$GO)$sign_VCG`, and `metadata(pbmcs$GO)$sign_all`.
```

`swap_pass` *Perform one-shot adjacent swapping for each element.*

Description

Perform one-shot adjacent swapping for each element.

Usage

```
swap_pass(listdata)
```

Arguments

`listdata`	A list of vector and integer.

Value

A List.

Examples

```
swap_pass(list(vec = c(1, 1, 0), cnt = 0))
```

Index