Package 'SingleCellAlleleExperiment'

November 12, 2026

Title S4 Class for Single Cell Data with Allele and Functional Levels for Immune Genes

Version 1.8.0

Description
Defines a S4 class that is based on SingleCellExperiment. In addition to the usual gene layer the object can also store data for immune genes such as HLAs, Igs and KIRs at allele and functional level.
The package is part of a workflow named single-cell ImmunoGenomic Diversity (scIGD), that firstly incorporates allele-aware quantification data for immune genes.
This new data can then be used with the here implemented data structure and functionalities for further data handling and data analysis.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Depends R (>= 4.4.0), SingleCellExperiment

Imports SummarizedExperiment, BiocParallel, DelayedArray, methods, utils, Matrix, S4Vectors, stats

Suggests scaeData, knitr, rmarkdown, BiocStyle, scran, scater, scuttle, ggplot2, patchwork, org.Hs.eg.db, AnnotationDbi, DropletUtils, testthat (>= 3.0.0)

biocViews DataRepresentation, Infrastructure, SingleCell, Transcriptomics, GeneExpression, Genetics, ImmunoOncology, DataImport

URL https://github.com/AGImkeller/SingleCellAlleleExperiment

BugReports https://github.com/AGImkeller/SingleCellAlleleExperiment/issues

VignetteBuilder knitr

Config/testthat/edition 3

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

git_branch RELEASE_3_23

git_last_commit 8c7ec7a

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-11-11

Author Jonas Schuck [aut, cre] (ORCID:
<https://orcid.org/0009-0003-5705-4579>),
Ahmad Al Ajami [aut] (ORCID: <https://orcid.org/0009-0006-5615-7447>),
Federico Marini [aut] (ORCID: <https://orcid.org/0000-0003-3252-7758>),
Katharina Imkeller [aut] (ORCID:
<https://orcid.org/0000-0002-5177-0852>)

Maintainer Jonas Schuck <jschuckdev@gmail.com>

Contents

alleles2genes . . . 2
check_valid_optional_package . . . 3
ext_rd . . . 4
find_allele_ids . . . 4
genes2functional . . . 5
get_agenes . . . 5
get_allelecounts . . . 6
get_knee_info . . . 6
get_ncbi_org . . . 7
get_nigenes . . . 7
read_allele_counts . . . 8
read_from_sparse_allele . . . 10
rowData<-,SingleCellAlleleExperiment,ANY-method . . . 11
rowData<-,SingleCellAlleleExperiment,NULL-method . . . 12
scae_subset . . . 13
scae_subset_alleles . . . 14
scae_subset_functional . . . 14
SingleCellAlleleExperiment-class . . . 15
SingleCellAlleleExperiment-misc . . . 17

Index **19**

`alleles2genes` *Building first new subassay for SingleCellAllelexperiment object*

Description

Internal function for the first assay extension used in the `SingleCellAlleleExperiment()` constructor, computing the first of the two new subassays that get appended to the quantification assay. This subassay contains the allele gene identifiers instead of the allele identifiers present in the raw data and sums up the expression counts of alleles that have the same allele gene identifiers.

Usage

```
alleles2genes(sce, lookup, exp_type, gene_symbols)
```

Arguments

sce	A SingleCellExperiment object.
lookup	A data.frame object containing the lookup table.
exp_type	Internal character string parameter that determines in which format the gene symbols in the input data are. Can be c("ENS","noENS")
gene_symbols	A logical parameter to decide whether to compute additional gene gene symbols in case the raw data only contains ENSEMBL gene identifiers.

Value

A SingleCellExperiment object

---

check_valid_optional_package

Check package installation for optional functionalities

Description

Check package installation for optional functionalities

Usage

```
check_valid_optional_package(log, gene_symbols)
```

Arguments

log	A logical parameter to decide if a logcounts assay should be computed based on library factors computed with scuttle::computeLibraryFactors().
gene_symbols	A logical parameter to decide whether to compute additional gene gene symbols in case the raw data only contains ENSEMBL gene identifiers.

Value

Error messages if cases are met

ext_rd *Extend rowData with new annotation columns*

Description

Extend rowData with new annotation columns

Usage

```
ext_rd(sce, exp_type, gene_symbols, verbose = FALSE)
```

Arguments

sce	A SingleCellExperiment object.
exp_type	Internal character string parameter that determines in which format the gene symbols in the input data are. Can be c("ENS","noENS")
gene_symbols	A logical parameter to decide whether to compute additional gene gene symbols in case the raw data only contains ENSEMBL gene identifiers.
verbose	A logical parameter to decide if runtime-messages should be shown during function execution. Use FALSE if no info runtime-messages should be shown (default), and TRUE for showing runtime-messages.

Value

A SingleCellExperiment object

find_allele_ids *Identify rows containing allele information*

Description

Internal function used in get_allelecounts() to subsample the quantification assay and only return the rows specifying allele-quantification information.

Usage

```
find_allele_ids(sce)
```

Arguments

sce	A SingleCellExperiment object.

Value

A SingleCellExperiment object

genes2functional *Building second new subassay for the SingleCellAlleleExperiment object*

Description

Internal function for the second assay extension used in the `SingleCellAlleleExperiment()` constructor, computing the second of the two new subassays that get appended to the quantification assay. This subassay contains the functional allele classes and sums up the expression counts of the allele genes that are in the same functional group.

Usage

```
genes2functional(sce, lookup, exp_type, gene_symbols)
```

Arguments

`sce`	A `SingleCellExperiment` object.
`lookup`	A data.frame object containing the lookup table.
`exp_type`	Internal character string parameter that determines in which format the gene symbols in the input data are. Can be `c("ENS","noENS")`
`gene_symbols`	A logical parameter to decide whether to compute additional gene gene symbols in case the raw data only contains ENSEMBL gene identifiers.

Value

A SingleCellExperiment object

get_agenes *Get immune gene rows*

Description

Getter function returning subsampled SCAE object with all rows containing immune gene information. These rows are identified by "I" in rowData(scae)$NI_I and "G" in rowData(scae)$Quant_type.

Usage

```
get_agenes(scae)
```

Arguments

`scae`	A `SingleCellAlleleExperiment` object.

Value

A SingleCellAlleleExperiment object.

get_allelecounts *Get Subassay with allele gene names and raw allele quantification*

Description

Internal function used to build a subassay containing counts from raw alleles. The rownames of this subassay are already translated to the corresponding immune gene identifier, which are extracted from the lookup table.

Usage

```
get_allelecounts(sce, lookup)
```

Arguments

sce	A SingleCellExperiment object.
lookup	A data.frame object containing the lookup table.

Value

A SingleCellExperiment object

get_knee_info *Knee plot info*

Description

Creates a knee plot information, ranking the barcodes according to their total UMI count. The information is later on passed to the `metadata(scae)[["knee_info"]]` slot.

Usage

```
get_knee_info(matrix, genes, barcodes)
```

Arguments

matrix	A sparse Matrix object containing the quantification data.
genes	A data.frame object containing gene identifiers.
barcodes	A data.frame object containing barcode identifiers.

Value

A list including a data.frame with barcode rank information, the corresponding knee and inflection point.

get_ncbi_org *Get NCBI genes using the org.HS.db package*

Description

Get NCBI genes using the org.HS.db package

Usage

```
get_ncbi_org(sce)
```

Arguments

sce	A SingleCellExperiment object.

Value

A list of character strings for gene names.

get_nigenes *Get non-immune rows*

Description

Getter function returning subsampled SCAE object with all rows containing non-immune gene information. These rows are identified by "NI" in rowData(scae)$NI_I and "G" in rowData(scae)$Quant_type.

Usage

```
get_nigenes(scae)
```

Arguments

scae	A SingleCellAlleleExperiment object.

Value

A SingleCellAlleleExperiment object.

`read_allele_counts` *Reading in allele quantification data into SingleCellAlleleExperiment object*

Description

Main read in function for reading in allele quantification data and loading the data into an `SingleCellAlleleExperiment` object.

Usage

```
read_allele_counts(
  samples_dir,
  sample_names = names(samples_dir),
  filter_mode = c("no", "yes", "custom"),
  lookup_file = lookup,
  barcode_file = "cells_x_genes.barcodes.txt",
  gene_file = "cells_x_genes.genes.txt",
  matrix_file = "cells_x_genes.mtx",
  filter_threshold = NULL,
  log = FALSE,
  gene_symbols = FALSE,
  verbose = FALSE,
  BPPARAM = BiocParallel::SerialParam()
)
```

Arguments

`samples_dir`	A character string determining the path to one directory containing input files.
`sample_names`	A character string for a sample identifier. Can be used to describe the used dataset or sample.
`filter_mode`	A vector containing three character strings that describe different options for filtering. The value `"yes"` uses the inflection point of the knee plot to filter out low-quality cells. The default value `"no"` performs filtering on a `threshold=0`. The value `"custom"` allows for setting a custom threshold in the `filter_threshold` parameter.
`lookup_file`	A character string determining the path to the lookup table.
`barcode_file`	A character string determining the name of the file containing the barcode identifiers.
`gene_file`	A character string determining the name of the file containing the feature identifiers.
`matrix_file`	A character string determining the name of the file containing the count matrix.
`filter_threshold`	An integer value used as a threshold for filtering low-quality barcodes/cells. Standard value is `NULL` when using `filter = c("yes", "no")`. Value must be provided when using `filter = "custom"`.

log	A logical parameter to decide if logcounts assay should be computed based on library factors computed with scuttle::computeLibraryFactors().
gene_symbols	A logical parameter to decide whether to compute additional gene gene symbols in case the raw data only contains ENSEMBL gene identifiers.
verbose	A logical parameter to decide if additional runtime-messages should be shown during function execution. Use FALSE if no info runtime-messages should be shown (default), and TRUE for showing runtime-messages.
BPPARAM	A BiocParallelParam object specifying how loading should be parallelized for multiple samples.

Details

The SingleCellAlleleExperiment data structure serves as a data representation for data generated with the scIGD workflow. This workflow allows for the quantification of expression and interactive exploration of donor-specific alleles of different immune genes and its

Input data are generated by the scIGD workflow is stored in a shared folder. Expected naming scheme of the files from the data generating method:

- quantification matrix: cells_x_genes.mtx
- barcode information: cells_x_genes.barcodes.txt
- feature information: cells_x_genes.genes.txt
- allele lookup table: lookup_table.csv

File identifiers can be specifically stated if renamed.

Optional features:

- Filtering: Used parameter is filter_mode. Default filtering is performed with a threshold=0 UMIs. filter_mode="yes" performs advanced filtering based on ranking the barcodes and infering a inflection point of a **knee plot**. Information regarding the knee plot is exported in the metadata(scae)[["knee_info"]] slot for later plotting (see vignette).
- Computing a logcount assay by normalizing the input data based on a sizeFactor method recommended for single-cell data. Used parameter is log=TRUE/FALSE.
- Computing additional gene symbols in case the input data only contains gene identifiers represented as Ensembl ids. Used parameter is gene_symbols=TRUE/FALSE.

Value

A SingleCellAlleleExperiment object.

See Also

SingleCellAlleleExperiment

Examples

```
example_data_5k <- scaeData::scaeDataGet(dataset="pbmc_5k")
lookup_name <- "pbmc_5k_lookup_table.csv"
lookup <- read.csv(system.file("extdata", lookup_name, package="scaeData"))


# preflight mode, default filtering with a threshold of 0 UMI counts
scae_preflight <- read_allele_counts(example_data_5k$dir,
                          sample_names="example_data",
                          filter_mode="no",
                          lookup_file=lookup,
                          barcode_file=example_data_5k$barcodes,
                          gene_file=example_data_5k$features,
                          matrix_file=example_data_5k$matrix,
                          filter_threshold=NULL)

scae_preflight

# automatic filtering mode, filtering out low-quality cells
# on the inflection point of the knee plot
#scae_filtered <- read_allele_counts(example_data_5k$dir,
#                          sample_names="example_data",
#                          filter_mode="yes",
#                          lookup_file=lookup,
#                          barcode_file=example_data_5k$barcodes,
#                          gene_file=example_data_5k$features,
#                          matrix_file=example_data_5k$matrix,
#                          filter_threshold=NULL,
#                          verbose=TRUE)

# scae_filtered

# custom filtering mode, setting up a custom filter threshold for filtering
# scae_custom_filter <- read_allele_counts(example_data_5k$dir,
#                          sample_names="example_data",
#                          filter_mode="custom",
#                          lookup_file=lookup,
#                          barcode_file=example_data_5k$barcodes,
#                          gene_file=example_data_5k$features,
#                          matrix_file=example_data_5k$matrix,
#                          filter_threshold=200)

# scae_custom_filter
```

`read_from_sparse_allele` *Reading in allele-aware quantification data*

Description

Internal function used in `read_allele_counts()` that reads in the data stated in the given directory path.

Usage

```
read_from_sparse_allele(path, barcode_file, gene_file, matrix_file)
```

Arguments

`path`	A character string determining the path to the directory containing the input files.
`barcode_file`	A character string determining the name of the file containing the sample-tag quantification data.
`gene_file`	A character string determining the name of the file containing the feature identifiers.
`matrix_file`	A character string determining the name of the file containing the count matrix.

Value

A list with three data.frames containing the input data information.

`rowData<-,SingleCellAlleleExperiment,ANY-method`

rowData setter for the SingleCellAlleleExperiment class

Description

Setter function for the rowData slot for the `SingleCellAlleleExperiment` class.

Usage

```
## S4 replacement method for signature 'SingleCellAlleleExperiment,ANY'
rowData(x) <- value
```

Arguments

`x`	A `SingleCellAlleleExperiment` object
`value`	Value of valid type and content (see validty.R)

Details

If you set `rowData(scae)<- NULL` the mandatory columns "NI_I" and "Quant_type" will be kept silently, setting all other columns to NULL.

If you want to change the content of the mandatory "NI_I" and "Quant_type" columns check the valid values:

- NI_I: c("NI" and "I") are valid values.
- Quant_type: c("A", "G" "F") are valid values.

Value

A `SingleCellAlleleExperiment` object

See Also

SingleCellAlleleExperiment

`rowData<-,SingleCellAlleleExperiment,NULL-method` *rowData-NULL-setter for the SingleCellAlleleExperiment class*

Description

Setter function for the rowData slot for the `SingleCellAlleleExperiment` class.

Usage

```
## S4 replacement method for signature 'SingleCellAlleleExperiment,NULL'
rowData(x) <- value
```

Arguments

`x`	A `SingleCellAlleleExperiment` object
`value`	NULL

Value

A `SingleCellAlleleExperiment` object

scae_subset *Subset SCAE object*

Description

Function used for subsetting the different layers stored in a SingleCellAlleleExperiment object. Valid subset values are: `subset=c("nonimmune", "alleles", "immune_genes", "functional_groups")`.

Usage

```
scae_subset(
  scae,
  subset = c("nonimmune", "alleles", "immune_genes", "functional_groups")
)
```

Arguments

scae	SCAE object
subset	Character string specifying a data layer. Valid values are `subset=c("nonimmune", "alleles", "immune_genes", "functional_groups")`.

Value

SCAE object

Examples

```
example_data_5k <- scaeData::scaeDataGet(dataset="pbmc_5k")
lookup_name <- "pbmc_5k_lookup_table.csv"
lookup <- read.csv(system.file("extdata", lookup_name, package="scaeData"))

scae <- read_allele_counts(example_data_5k$dir,
                           sample_names="example_data_wta",
                           filter_mode="no",
                           lookup_file=lookup,
                           barcode_file=example_data_5k$barcodes,
                           gene_file=example_data_5k$features,
                           matrix_file=example_data_5k$matrix,
                           filter_threshold=0,
                           verbose=TRUE)

scae

scae_nonimmune_subset <- scae_subset(scae, subset="nonimmune")
scae_nonimmune_subset

scae_alleles_subset <- scae_subset(scae, subset="alleles")
scae_alleles_subset
```

```
scae_immune_genes_subset <- scae_subset(scae, subset="immune_genes")
scae_immune_genes_subset

scae_functional_groups_subset <- scae_subset(scae, subset="functional_groups")
scae_functional_groups_subset
```

`scae_subset_alleles` *Get allele rows*

Description

Getter function returning subsampled SCAE object with all rows containing raw allele information. These rows are identified by "I" in rowData(scae)$NI_I and "A" in rowData(scae)$Quant_type.

Usage

```
scae_subset_alleles(scae)
```

Arguments

scae	A SingleCellAlleleExperiment object.

Value

A SingleCellAlleleExperiment object.

`scae_subset_functional` *Get functional class rows*

Description

Getter function returning subsampled SCAE object with all rows containing functional class information. These rows are identified by "I" in rowData(scae)$NI_I and "F" in rowData(scae)$Quant_type.

Usage

```
scae_subset_functional(scae)
```

Arguments

scae	A SingleCellAlleleExperiment object.

Value

A SingleCellAlleleExperiment object.

SingleCellAlleleExperiment-class
The SingleCellAlleleExperiment class

Description

The SingleCellAlleleExperiment class is a comprehensive multi-layer data structure, enabling the representatino of immune genes at specific levels, including alleles, genes and groups of functionally similar genes. This data representation allows data handling and data analysis across these immunological relevant, different layers of annotation.

Usage

```
SingleCellAlleleExperiment(
  ...,
  lookup,
  metadata = NULL,
  threshold = 0,
  exp_type = "ENS",
  log = TRUE,
  gene_symbols = FALSE,
  verbose = FALSE
)
```

Arguments

`...`	Arguments passed to the `SingleCellExperiment` constructor to fill the slots of the SCE-class.
`lookup`	A data.frame object containing the lookup table.
`metadata`	A list containing a dataframe and two integer values of information regarding plotting a knee plot for quality control. This parameter is linked to `filter_mode="yes"` in the `read_allele_counts()` function.
`threshold`	An integer value used as a threshold for filtering low-quality barcodes/cells.
`exp_type`	Internal character string parameter that determines in which format the gene symbols in the input data are. Can be `c("ENS","noENS")`
`log`	A logical parameter which determines if the user wants to compute the `logcounts` assay.
`gene_symbols`	A logical parameter to decide whether to compute additional gene gene symbols in case the raw data only contains ENSEMBL gene identifiers.
`verbose`	A logical parameter to decide if runtime-messages should be shown during function execution. Use `FALSE` if no info runtime-messages should be shown (default), and `TRUE` for showing runtime-messages.

Details

The SingleCellAlleleExperiment class builds upon and extends the data representation that can be facilitated using a `SingleCellExperiment` object.

The Constructor `SingleCellAlleleExperiment()` can be used on its own, if raw data is processed accordingly (see examples) OR in a more convenient way using this packages read in function `read_allele_counts()`

A getter function `scae_subset()` allows to subset the object according to the newly implemented layers.

In this class, similar to the `SingleCellExperiment` class, rows should represent genomic features (including immune genes, represented as allele information), while columns represent single cells/barcodes.

The SingleCellAlleleExperiment data structure serves as a data representation for data generated with the `scIGD` workflow. This workflow allows for the quantification of expression and interactive exploration of donor-specific alleles of different immune genes and its

Value

A SingleCellAlleleExperiment object.

See Also

`read_allele_counts()`

`scae_subset()`

Examples

```
##-If you want to use the Constructor on its own, some preprocessing is
##-necessary to bring the data in proper format
##-Here, we use an example dataset found in in the `scaeData` package.

##-Find an alternative and recommended read in below as a second example

example_data_5k <- scaeData::scaeDataGet(dataset="pbmc_5k")
lookup_name <- "pbmc_5k_lookup_table.csv"
lookup <- read.csv(system.file("extdata", lookup_name, package="scaeData"))

barcode_loc <- file.path(example_data_5k$dir, example_data_5k$barcodes)
feature_loc <- file.path(example_data_5k$dir, example_data_5k$features)
matrix_loc  <- file.path(example_data_5k$dir, example_data_5k$matrix)

feature_info <- utils::read.delim(feature_loc, header=FALSE)
cell_names   <- utils::read.csv(barcode_loc, sep="", header=FALSE)
mat <- t(Matrix::readMM(matrix_loc))

##-Prepare input data
colnames(feature_info) <- "Ensembl_ID"
sample_names <- "pbmc_5k"
sparse_mat <- as(mat, "CsparseMatrix")
```

```
##--colData
cell_info_list <- S4Vectors::DataFrame(Sample=rep(sample_names,
                                                  length(cell_names)),
                                       Barcode=cell_names$V1,
                                       row.names=NULL)
##--rowData and count matrix
rownames(feature_info) <- feature_info[,1]
cnames <- cell_info_list$Barcode
colnames(sparse_mat) <- cnames

scae <- SingleCellAlleleExperiment(assays=list(counts=sparse_mat),
                                   rowData=feature_info,
                                   colData=cell_info_list,
                                   lookup=lookup,
                                   verbose=TRUE)

scae

##-OR, use the read in function `read_allele_counts()` ![RECOMMENDED]!!
##-Find more examples in its documentation using `?read_allele_counts`

# scae_2 <- read_allele_counts(example_data_5k$dir,
#                              sample_names="example_data",
#                              filter_mode="no",
#                              lookup_file=lookup,
#                              barcode_file=example_data_5k$barcodes,
#                              gene_file=example_data_5k$features,
#                              matrix_file=example_data_5k$matrix,
#                              verbose=TRUE)

# scae_2
```

`SingleCellAlleleExperiment-misc` *Miscellaneous SingleCellAlleleExperiment methods*

Description

Miscellaneous methods for the `SingleCellAlleleExperiment` class and its descendants that do not fit into any other documentation category such as, for example, show methods.

Usage

```
## S4 method for signature 'SingleCellAlleleExperiment'
show(object)
```

Arguments

`object`	a `SingleCellAlleleExperiment` object

Value

Returns NULL

Index

.scae
        (SingleCellAlleleExperiment-class), 15

alleles2genes, 2

check_valid_optional_package, 3

ext_rd, 4

find_allele_ids, 4

genes2functional, 5
get_agenes, 5
get_allelecounts, 6
get_knee_info, 6
get_ncbi_org, 7
get_nigenes, 7

Matrix, *6*

read_allele_counts, 8
read_allele_counts(), *16*
read_from_sparse_allele, 10
rowData<-,SingleCellAlleleExperiment,ANY-method, 11
rowData<-,SingleCellAlleleExperiment,NULL-method, 12

scae_subset, 13
scae_subset(), *16*
scae_subset_alleles, 14
scae_subset_functional, 14
show,SingleCellAlleleExperiment-method
        (SingleCellAlleleExperiment-misc), 17
SingleCellAlleleExperiment, *5*, *7*, *9*, *11*, *12*, *14*, *17*
SingleCellAlleleExperiment
        (SingleCellAlleleExperiment-class), 15
SingleCellAlleleExperiment-class, 15
SingleCellAlleleExperiment-misc, 17
SingleCellExperiment, *3–7*, *15*, *16*