Package 'scafari'

April 7, 2026

Type Package

Title Analysis of scDNA-seq data

Version 1.1.0

Description Scafari is a Shiny application designed for the analysis of single-cell DNA sequencing (scDNA-seq) data provided in .h5 file format. The analysis process is structured into the four key steps ``Sequencing'', ``Panel'', ``Variants'', and ``Explore Variants''. It supports various analyses and visualizations.

Depends R (>= 4.5.0)

License LGPL-3

Encoding UTF-8

Imports magrittr, shiny, shinycssloaders, DT, dplyr, waiter, ggplot2, tibble, stringr, reshape2, shinyjs, shinyBS, shinycustomloader, factoextra, markdown, plotly, ggbio, GenomicRanges, rhdf5, ComplexHeatmap, biomaRt, org.Hs.eg.db, SummarizedExperiment, SingleCellExperiment, S4Vectors, parallel, httr, jsonlite, scales, tidyr, txdbmaker, circlize, R.utils, dbscan, igraph, RANN

VignetteBuilder knitr

Suggests knitr, rmarkdown, BiocStyle, testthat (>= 3.0.0)

biocViews Software, ShinyApps, SingleCell, Sequencing

BugReports https://github.com/sophiewind/scafari/issues

URL https://github.com/sophiewind/scafari

RoxygenNote 7.3.2

Config/testthat/edition 3

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

git_branch devel

git_last_commit 0bee2a7

git_last_commit_date 2025-10-29

Repository Bioconductor 3.23

Date/Publication 2026-04-06

Author Sophie Wind [aut, cre] (ORCID: <https://orcid.org/0009-0002-1174-8201>)

Maintainer Sophie Wind <sophie.wind@uni-muenster.de>

Contents

`annotateAmplicons` *Function: annotateAmplicons This function takes a SingleCellExperiment object as input and annotates the stored amplicons.*

Description

Function: annotateAmplicons This function takes a SingleCellExperiment object as input and annotates the stored amplicons.

Usage

```
annotateAmplicons(sce, known.canon, shiny = FALSE)
```

Arguments

sce	SingleCellExperiment object containing the single-cell data.
known.canon	Path to jnown canonicals (see vignette)
shiny	If TRUE messages are shown

Value

A dataframe containing annotated amplicons.

Examples

```
# Assume `sce` is a SingleCellExperiment object with a 'counts' assay
sce_filtered <- readRDS(system.file("extdata", "sce_filtered_demo.rds",
    package = "scafari"
))

annotated <- annotateAmplicons(
    sce_filtered,
    system.file("extdata", "UCSC_hg19_knownCanonical_mock.txt",
        package = "scafari"
    )
)
```

---

annotateVariants *Function: annotateVariants This function takes a SingleCellExperiment object as input and performs variant annotation.*

Description

Function: annotateVariants This function takes a SingleCellExperiment object as input and performs variant annotation.

Usage

```
annotateVariants(sce, shiny = FALSE, max.var = 50)
```

Arguments

sce	SingleCellExperiment object containing the single-cell data to be annotated.
shiny	A logical flag indicating whether the function is being run in a Shiny application context. Default is FALSE.
max.var	Maximum number of variants to annotate. By default this is 50 to avoid long runtime.

Value

The function returns an annotated SingleCellExperiment object.

References

https://missionbio.github.io/mosaic/, https://github.com/rachelgriffard/optima

Examples

```
# Assume `sce` is a SingleCellExperiment object with variants in altExp()
sce_filtered <- readRDS(system.file("extdata", "sce_filtered_demo.rds",
    package = "scafari"
))
sce <- annotateVariants(sce_filtered, shiny = FALSE)
```

---

checkH5 *Check h5*

Description

Check h5

Usage

```
checkH5(h5_file)
```

Arguments

h5_file	and path to the h5 file

Value

boolean if h5 is valid

---

check_MBAPI *Check if MissionBio is accessible*

Description

Check if MissionBio is accessible

Usage

```
check_MBAPI()
```

Value

boolean if MissionBio API is currently available

clusterVariantSelection

Function: clusterVariantSelection This function takes selected variants and performs clustering on them.

Description

Function: clusterVariantSelection This function takes selected variants and performs clustering on them.

Usage

```
clusterVariantSelection(
  sce,
  variants.of.interest,
  n.clust,
  method = "k-means",
  eps.value = 0.2,
  resolution = NULL,
  min.pts = NULL
)
```

Arguments

sce	A SingleCellExperiment object containing the single-cell data on which clustering will be performed.
variants.of.interest	A vector or list specifying the variants of interest to be selected for clustering.
n.clust	An integer specifying the number of clusters.
method	Clustering method. Either k-means, dbscan or leiden.
eps.value	Size (radius) of the epsilon neighborhood. Can be omitted if x is a frNN object.
resolution	The resolution parameter to use. Higher resolutions lead to more smaller communities, while lower resolutions lead to fewer larger communities.
min.pts	Number of minimum points required in the eps neighborhood for core points (including the point itself). By default cell number/100.

Value

A list with clustering results and a ggplot-object.

References

https://cran.r-project.org/web/packages/dbscan/readme/ README.html#ref-hahsler2019dbscan

Examples

```
# Assume `sce` is a SingleCellExperiment object with variants in altExp()
sce_filtered <- readRDS(system.file("extdata", "sce_filtered_demo.rds",
    package = "scafari"
))
clusterplot <- clusterVariantSelection(
    sce = sce_filtered,
    variants.of.interest = c(
        "FLT3:chr13:28610183:A/G",
        "KIT:chr4:55599436:T/C",
        "TP53:chr17:7577427:G/A",
        "TET2:chr4:106158216:G/A"
    ),
    n.clust = 4
)
```

`filterVariants` *Function: filterVariants ———————————- This function takes a SingleCellExperiment object as input and performs variant filtering*

Description

Function: filterVariants ———————————- This function takes a SingleCellExperiment object as input and performs variant filtering

Usage

```
filterVariants(
  depth.threshold = numeric(),
  genotype.quality.threshold = numeric(),
  vaf.ref = numeric(),
  vaf.het = numeric(),
  vaf.hom = numeric(),
  min.cell = numeric(),
  min.mut.cell = numeric(),
  se.var,
  sce,
  shiny = FALSE
)
```

Arguments

`depth.threshold`
: A numeric value specifying the minimum read depth required.

`genotype.quality.threshold`
: A numeric value specifying the minimum genotype quality score.

vaf.ref	A numeric value specifying the variant allele frequency threshold for wild-type alleles.
vaf.het	A numeric value specifying the variant allele frequency threshold for heterozygous variants.
vaf.hom	A numeric value specifying the variant allele frequency threshold for homozygous variants.
min.cell	A numeric value indicating the minimum number of cells with a genotype other than "missing".
min.mut.cell	A numeric value indicating the minimum number of mutated (genotype either "homozygous" or "heterozygous") cells.
se.var	The SummarizedExperiment object containing variant data which will be filtered.
sce	SingleCellExperiment object containing the single-cell data TODO.
shiny	A logical flag indicating whether the function is being run in a Shiny application context. Default is FALSE.

Value

A list containing the following elements:

vaf.matrix.filtered Variant allele frequencies after filtering.

genotype.matrix.filtered Genotype information after filtering.

read.counts.df.norm.filtered Normalized read counts for variants retained after filtering.

variant.ids.filtered A vector of the variant IDs that were retained after filtering.

genoqual.matrix.filtered Genotype qualities for variants retained after filtering.

cells.keep A vector of cell identifiers for those cells retaine d after filtering.

References

https://missionbio.github.io/mosaic/, https://github.com/rachelgriffard/optima

Examples

```
library(SummarizedExperiment)
h5_file_path <- system.file("extdata", "demo.h5", package = "scafari")
h5 <- h5ToSce(h5_file_path)
sce <- h5$sce_amp
se.var <- h5$se_var
sce <- normalizeReadCounts(sce = sce)
filteres <- filterVariants(
    depth.threshold = 10,
    genotype.quality.threshold = 30,
    vaf.ref = 5,
    vaf.het = 35,
    vaf.hom = 95,
    min.cell = 50,
    min.mut.cell = 1,
```

```
    se.var = se.var,
    sce = sce,
    shiny = FALSE
)
se.f <- SummarizedExperiment(
    assays = list(
        VAF = t(filteres$vaf.matrix.filtered),
        Genotype = t(filteres$genotype.matrix.filtered),
        Genoqual = t(filteres$genoqual.matrix.filtered)
    ),
    rowData = filteres$variant.ids.filtered,
    colData = filteres$cells.keep
)

# Filter out cells in sce object
# Find the indices of the columns to keep
indices_to_keep <- match(filteres$cells.keep,
    SummarizedExperiment::colData(sce)[[1]],
    nomatch = 0
)

# Subset the SCE using these indices
sce_filtered <- sce[, indices_to_keep]
```

`gg_color_hue` *Default ggplot colors*

Description

Default ggplot colors

Usage

```
gg_color_hue(n)
```

Value

gg colors for plotting

`h5ToSce` *Function: h5ToSce This function takes the path of an h5 file and reads this into an object of the SingleCellExperiment class.*

Description

Function: h5ToSce This function takes the path of an h5 file and reads this into an object of the SingleCellExperiment class.

Usage

```
h5ToSce(h5_file)
```

Arguments

h5_file	path of an h5 file

Value

A list containing the following elements:

sce_amp SingleCellExperiment class object containing read count information.

se_var SummarizedExperiment class object containing variant information..

A list with the SingleCellExperiment and the SummarizedExperiment object representing the amplicon and the variant analysis.

sce_amp SingleCellExperiment with amplicon experiment.

se_var SummarizedExperiment with variants eexperiment

Examples

```
h5_file_path <- system.file("extdata", "demo.h5", package = "scafari")

# Read the h5ToSce using readH5File
result <- h5ToSce(h5_file_path)

# Display the result
result
```

---

launchScafariShiny *Launch Scafari Shiny Application*

Description

Launches the Scafari Shiny application for data visualization.

Usage

```
launchScafariShiny()
```

Value

shiny app

Examples

```
if (interactive()) {
    launchScafariShiny()
}
```

`logLogPlot` *Plot a log-log plot for read counts*

Description

This function generates a log-log plot of read counts using the data from a 'SingleCellExperiment' object.

Usage

```
logLogPlot(sce)
```

Arguments

`sce`	A SingleCellExperiment object that contains read count data to be plotted. The read counts are extracted from an associated h5 file or similar data structure within the object.

Value

A ggplot object representing the log-log plot of read counts.

Examples

```
# Assume `sce` is a SingleCellExperiment object with a 'counts' assay
h5_file_path <- system.file("extdata", "demo.h5", package = "scafari")
h5 <- h5ToSce(h5_file_path)
sce <- h5$sce_amp
plot <- logLogPlot(sce)
print(plot)
```

normalizeReadCounts *This function normalizes the read counts contained within a 'SingleCellExperiment' object.*

Description

This function normalizes the read counts contained within a 'SingleCellExperiment' object.

Usage

```
normalizeReadCounts(sce)
```

Arguments

sce	A SingleCellExperiment object that includes the assay data with read counts to be normalized. The metadata within the object may also be utilized for normalization purposes.

Value

SingleCellExperiment object with normalized read counts.

References

https://missionbio.github.io/mosaic/, https://github.com/rachelgriffard/optima

Examples

```
# Assume `sce` is a SingleCellExperiment object with 'counts' assay.
h5_file_path <- system.file("extdata", "demo.h5", package = "scafari")
h5 <- h5ToSce(h5_file_path)
sce <- h5$sce_amp
sce <- normalizeReadCounts(sce)
```

plotAmpliconDistribution *Plot Distribution of Amplicons*

Description

This function generates a plot to visualize the distribution of amplicons within a 'SingleCellExperiment' object.

Usage

```
plotAmpliconDistribution(sce)
```

Arguments

sce	A SingleCellExperiment object that contains the assay data, including amplicon information to be plotted.

Value

A ggplot object representing the distribution of amplicons.

Examples

```
# Assume `sce` is a SingleCellExperiment object with 'counts' assay.
h5_file_path <- system.file("extdata", "demo.h5", package = "scafari")
h5 <- h5ToSce(h5_file_path)
sce <- h5$sce_amp
plotAmpliconDistribution(sce)
```

---

plotClusterGenotype *Plot Genotype Clusters*

Description

This function generates a plot to visualize genotype in clusters based on selected variants of interest.

Usage

```
plotClusterGenotype(sce, variants.of.interest, gg.clust)
```

Arguments

sce	A SingleCellExperiment object containing the relevant data.
variants.of.interest	A vector specifying the variants of interest.
gg.clust	An object containing clustering information.

Value

A ggplot object that visually represents the clustering of genotypes based on the specified variants and clustering information.

Examples

```
# Assume `sce` is a SingleCellExperiment object with variants in altExp()
# and clusterplot is the output of clusterVariantSleection().
sce_filtered <- readRDS(system.file("extdata", "sce_filtered_demo.rds",
    package = "scafari"
))
clusterplot <- readRDS(system.file("extdata", "clusterplot.rds",
    package = "scafari"
))
plotClusterGenotype(
    sce = sce_filtered,
    variants.of.interest = c(
        "FLT3:chr13:28610183:A/G",
        "KIT:chr4:55599436:T/C",
        "TP53:chr17:7577427:G/A",
        "TET2:chr4:106158216:G/A"
    ),
    gg.clust = clusterplot$clusterplot
)
```

`plotClusterVAF` *plot Cluster VAF This function generates a plot to visualize variant allele frequency (VAF) in clusters based on selected variants of interest.*

Description

plot Cluster VAF This function generates a plot to visualize variant allele frequency (VAF) in clusters based on selected variants of interest.

Usage

```
plotClusterVAF(sce, variants.of.interest, gg.clust)
```

Arguments

`sce`	A SingleCellExperiment object containing the relevant data.
`variants.of.interest`	A vector specifying the variants of interest.
`gg.clust`	An object containing clustering information.

Value

A ggplot object that visually represents the VAF in the clusters.

Examples

```
# Assume `sce` is a SingleCellExperiment object with variants in altExp() and
# clusterplot is the output of clusterVariantSleection().
sce_filtered <- readRDS(system.file("extdata", "sce_filtered_demo.rds",
    package = "scafari"
))
clusterplot <- readRDS(system.file("extdata", "clusterplot.rds",
    package = "scafari"
))
plotClusterVAF(
    sce = sce_filtered,
    variants.of.interest = c(
        "FLT3:chr13:28610183:A/G",
        "KIT:chr4:55599436:T/C",
        "TP53:chr17:7577427:G/A",
        "TET2:chr4:106158216:G/A"
    ),
    gg.clust = clusterplot$clusterplot
)
```

`plotClusterVAFMap` *plot Cluster VAF Map This function generates a plot to visualize variant allele frequency (VAF) in clusters based on selected variants of interest with clusters in the background.*

Description

plot Cluster VAF Map This function generates a plot to visualize variant allele frequency (VAF) in clusters based on selected variants of interest with clusters in the background.

Usage

```
plotClusterVAFMap(sce, variants.of.interest, gg.clust)
```

Arguments

`sce`	A SingleCellExperiment object containing the relevant data.
`variants.of.interest`	A vector specifying the variants of interest.
`gg.clust`	An object containing clustering information.

Value

A ggplot object that visually represents the VAF in the clusters with clusters in the background.

Examples

```
# Assume `sce` is a SingleCellExperiment object with variants in altExp() and
# clusterplot is the output of clusterVariantSleection().
sce_filtered <- readRDS(system.file("extdata", "sce_filtered_demo.rds",
    package = "scafari"
))
clusterplot <- readRDS(system.file("extdata", "clusterplot.rds",
    package = "scafari"
))
plotClusterVAFMap(
    sce = sce_filtered,
    variants.of.interest = c(
        "FLT3:chr13:28610183:A/G",
        "KIT:chr4:55599436:T/C",
        "TP53:chr17:7577427:G/A",
        "TET2:chr4:106158216:G/A"
    ),
    gg.clust = clusterplot$clusterplot
)
```

`plotElbow` *This function takes a SingleCellExperiment object and variants of interest as input and plots an elbow plot to perform k-means later.*

Description

This function takes a SingleCellExperiment object and variants of interest as input and plots an elbow plot to perform k-means later.

Usage

```
plotElbow(sce, variants.of.interest)
```

Arguments

`sce`	A SingleCellExperiment object containing the relevant data.
`variants.of.interest`	A vector specifying the variants of interest.

Value

ggplot object with elbow plot.

Examples

```
# Assume `sce` is a SingleCellExperiment object with variants in altExp().
sce_filtered <- readRDS(system.file("extdata", "sce_filtered_demo.rds",
package = "scafari"))
plotElbow(
    sce = sce_filtered,
    variants.of.interest = c(
        "FLT3:chr13:28610183:A/G",
        "KIT:chr4:55599436:T/C",
        "TP53:chr17:7577427:G/A",
        "TET2:chr4:106158216:G/A"
    )
)
```

`plotGenotypequalityPerGenotype`

Plot Genotype Quality per Genotype

Description

This function generates a plot to assess the quality of genotypes within a 'SingleCellExperiment' object. It uses the 'genotype_quality' assay or any relevant assay to visualize the distribution of genotype quality metrics across different genotypes.

Usage

```
plotGenotypequalityPerGenotype(sce)
```

Arguments

`sce`	A 'SingleCellExperiment' object containing the single-cell data. This object should include a 'genotype_quality' assay or similar data which provides quality metrics for each genotype.

Value

A 'ggplot' object visualizing the distribution of genotype quality across different genotypes.

Examples

```
# Assume `sce` is a SingleCellExperiment object with 'genotype_quality'
# assay.
sce_filtered <- readRDS(system.file("extdata", "sce_filtered_demo.rds",
    package = "scafari"
))
genotype_quality_plot <- plotGenotypequalityPerGenotype(sce_filtered)
print(genotype_quality_plot)
```

plotNormalizedReadCounts *Plot normalized read counts*

Description

This function plots the normalize read counts in a bar chart.

Usage

```
plotNormalizedReadCounts(sce)
```

Arguments

sce	A SingleCellExperiment object containing the relevant data.

Value

ggplot object visualizing the normalized read counts in a bar chart.

Examples

```
# Assume `sce` is a SingleCellExperiment object with 'counts' assay.
h5_file_path <- system.file("extdata", "demo.h5", package = "scafari")
h5 <- h5ToSce(h5_file_path)
sce <- h5$sce_amp
sce <- normalizeReadCounts(sce = sce)
plotNormalizedReadCounts(sce)
```

plotPanelUniformity *Plot Panel Uniformity*

Description

This function generates a plot to assess the uniformity of panel reads in a 'SingleCellExperiment' object. It uses read counts stored in the 'counts' assay to visualize the distribution and variability of reads.

Usage

```
plotPanelUniformity(sce, interactive = FALSE)
```

Arguments

sce	A 'SingleCellExperiment' object that contains single-cell data with a 'counts' assay. This object should be pre-processed to ensure the data is appropriate for uniformity analysis.
interactive	in interactive mode an plotly is returned.

Value

A 'ggplot' object visualizing the uniformity of panel reads.

Examples

```
# Assume `sce` is a SingleCellExperiment object with 'counts' assay.
h5_file_path <- system.file("extdata", "demo.h5", package = "scafari")
h5 <- h5ToSce(h5_file_path)
sce <- h5$sce_amp
sce <- normalizeReadCounts(sce = sce)
uniformity_plot <- plotPanelUniformity(sce)
print(uniformity_plot)
```

---

plotVariantHeatmap *Plot Variant Heatmap*

Description

This function generates a heatmap to visualize variant data within a 'SingleCellExperiment' object. The heatmap provides insights into the distribution and frequency of variants across different samples or cells.

Usage

```
plotVariantHeatmap(sce)
```

Arguments

sce	A 'SingleCellExperiment' object containing single-cell variant data. The object should include an assay that holds variant information suitable for visualization in a heatmap.

Value

A 'ggplot' object representing the heatmap of variant frequencies or presence across cells or groups.

Examples

```
# Assume `sce` is a SingleCellExperiment object with an appropriate variant
# assay.
sce_filtered <- readRDS(system.file("extdata", "sce_filtered_demo.rds",
    package = "scafari"
))
variant_heatmap <- plotVariantHeatmap(sce_filtered)
print(variant_heatmap)
```

`scafari` *scafari: analyzing scDNA-seq data*

Description

scafari is an R Bioconductor package for single-cell DNA-seq (scDNA-seq) analysis.

Details

scafari works on .h5 files, the standard output of the Tapestri pipeline. It offers easy-to-use data quality control as well as explorative variant analyses and visualization for scDNA-seq data.

Main Functions

- `h5ToSce()`: Reads in .h5 files and writes them into a SingleCellExperiment object.
- `normalizeReadCounts()`: Normalizes the read counts.
- `annotateAmplicons()`: Annotates the amplicons present in the .h5 file.
- `filterVariants()`: Filters the variants present in the .h5 file.
- `annotateVariants()`: Annotates the filtered variants.
- `clusterVariantSelection()`: Clusteres variants of special interest.

Installation

To install this package, use: `BiocManager::install("scafari")`

Author(s)

Maintainer: Sophie Wind <sophie.wind@uni-muenster.de> (ORCID)

See Also

Useful links:

- https://github.com/sophiewind/scafari
- Report bugs at https://github.com/sophiewind/scafari/issues

theme_default *Default ggplot scheme*

Description

Default ggplot scheme

Usage

```
theme_default()
```

Value

gg color scheme

theme_shiny *Default shiny theme*

Description

Default shiny theme

Usage

```
theme_shiny()
```

Value

gg color scheme

Index