Package 'airpart'

November 12, 2026

Title Differential cell-type-specific allelic imbalance

Version 1.21.0

Description Airpart identifies sets of genes displaying differential cell-type-specific allelic imbalance across cell types or states, utilizing single-cell allelic counts. It makes use of a generalized fused lasso with binomial observations of allelic counts to partition cell types by their allelic imbalance. Alternatively, a nonparametric method for partitioning cell types is offered. The package includes a number of visualizations and quality control functions for examining single cell allelic imbalance datasets.

License GPL-2

Depends R (>= 4.1)

Imports SingleCellExperiment, SummarizedExperiment, S4Vectors, scater, stats, smurf, apeglm (>= 1.13.3), emdbook, mclust, clue, dynamicTreeCut, matrixStats, dplyr, plyr, ggplot2, ComplexHeatmap, forestplot, RColorBrewer, rlang, lpSolve, grid, grDevices, graphics, utils, pbapply

Suggests knitr, rmarkdown, roxygen2 (>= 6.0.0), testthat (>= 3.0.0), gplots, tidyr

VignetteBuilder knitr

biocViews SingleCell, RNASeq, ATACSeq, ChIPSeq, Sequencing, GeneRegulation, GeneExpression, Transcription, TranscriptomeVariant, CellBiology, FunctionalGenomics, DifferentialExpression, GraphAndNetwork, Regression, Clustering, QualityControl

Encoding UTF-8

BugReports https://github.com/Wancen/airpart/issues

URL https://github.com/Wancen/airpart

RoxygenNote 7.2.3

Config/testthat/edition 3

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

git_branch devel

git_last_commit 8e271c3

git_last_commit_date 2026-04-28

Repository Bioconductor 3.24

Date/Publication 2026-11-12

Author Wancen Mu [aut, cre] (ORCID: <https://orcid.org/0000-0002-5061-7581>), Michael Love [aut, ctb] (ORCID: <https://orcid.org/0000-0001-8401-0545>)

Maintainer Wancen Mu <wancen@live.unc.edu>

Contents

allelicRatio 2
cellQC 3
consensusPart 5
estDisp 5
extractResult 6
featureQC 7
fusedLasso 8
geneCluster 10
makeForest 11
makeHeatmap 12
makeOffByOneAdjMat 13
makeSimulatedData 14
makeStep 15
makeViolin 16
preprocess 16
summaryAllelicRatio 17
wilcoxExt 18

Index **20**

`allelicRatio` *Fit beta-binomial across cell types*

Description

This function performs additional inference on the allelic ratio across cell types, giving posterior mean and credible intervals per cell type. A Cauchy prior is centered for each cell type on the allelic ratio from the fused lasso across all genes in the gene cluster (or using a weighted means if the fused lasso is not provided).

Usage

```
allelicRatio(sce, formula, nogroup = FALSE, level = 0.95, DAItest = FALSE, ...)
```

Arguments

sce	A SingleCellExperiment containing assays ("ratio", "counts") and colData ("x", "part")
formula	The same formula object used in fusedLasso or used in ns when want to detect smooth continuous-axis estimates. eg. formula <- ratio ~ ns(x, df = 5)
nogroup	Indicate whether there is previous group step. Default is FALSE represents either Generalized fused lasso or nonparametric method is used to derive partition. nogroup == TRUE means the hypothesis that groups of cell types sharing a common regulatory context is not valid
level	the level of credible interval (default is 0.95)
DAItest	Indicate whether to do Likelihood Ratio test on differential allelic imbalance(DAI) or equivalent to heterogeneity.
...	arguments to pass to apeglm functions

Value

posterior mean ("ar") for allelic ratio estimate is returned in the rowData for each cell type, as well as the "s" value, "fsr" false sign rate and credible interval ("lower" and "upper"). One can use "fsr" < 0.005 or credible intervals contain 0.5 or not for AI test significance. "p.value" shows DAI test result and "adj.p.value" is false discovery rate corrected p values.

Examples

```
sce <- makeSimulatedData()
sce <- preprocess(sce)
sce <- geneCluster(sce, G = seq_len(4))
sce_sub <- wilcoxExt(sce, genecluster = 1)
sce_sub <- allelicRatio(sce_sub, DAItest = TRUE)
```

---

cellQC *Quality control on cells*

Description

Quality control on cells

Usage

```
cellQC(
  sce,
  spike,
  threshold = 0,
  mad_sum = 5,
  mad_detected = 3,
  mad_spikegenes = 5
)
```

Arguments

sce	SingleCellExperiment with counts and ratio
spike	the character name of spike genes. If missing, spikePercent will all be zero and filter_spike will be false.
threshold	A numeric scalar specifying the threshold above which a gene is considered to be detected.
mad_sum	A numeric scalar specifying exceed how many median absolute deviations from the median log10-counts a cell is considered to be filtered out. Default is 5.
mad_detected	A numeric scalar specifying exceed how many median absolute deviations from the median detected features a cell is considered to be filtered out. Default is 5.
mad_spikegenes	A numeric scalar specifying exceed how many median absolute deviations from the median spike genes expression percentage a cell is considered to be filtered out. Default is 5.

Value

A DataFrame of QC statistics includes

- sum the sum of counts of each cell
- detected the number of features above threshold
- spikePercent the percentage of counts assignes to spike genes
- filter_sum indicate whether log10-counts within mad_sum median absolute deviations from the median log10-counts for the dataset
- filter_detected indicate whether features detected by this cell within mad_detected median absolute deviations from the median detected features for the dataset
- filter_spike indicate whether percentage expressed by spike genes within mad_spikegenes median absolute deviations from the median spike genes expression percentage for the dataset

Examples

```
sce <- makeSimulatedData()
sce <- preprocess(sce)
cellQCmetrics <- cellQC(sce)
keep_cell <- (
  cellQCmetrics$filter_sum | # sufficient features (genes)
    # sufficient molecules counted
    cellQCmetrics$filter_detected |
    # sufficient features expressed compared to spike genes
    cellQCmetrics$filter_spike
)
sce <- sce[, keep_cell]

# or manually setting threshold
cellQCmetrics <- cellQC(sce,
  spike = "Ercc",
  mad_detected = 4, mad_spikegenes = 4
)
keep_cell <- (
  cellQCmetrics$sum > 2.4 |
    cellQCmetrics$detected > 110
)
```

consensusPart *Consensus Partitions*

Description

Derive consensus partitions of an ensemble fused lasso partitions.

Usage

```
consensusPart(sce)
```

Arguments

sce	SingleCellExperiment

Value

A matrix grouping factor partition is replaced in metadata. Consensus Partation also stored in colData"part".

Examples

```
library(smurf)
sce <- makeSimulatedData()
sce <- preprocess(sce)
sce <- geneCluster(sce, G = 1:4)
f <- ratio ~ p(x, pen = "gflasso") # formula for the GFL
sce_sub <- fusedLasso(sce,
  formula = f, model = "binomial", genecluster = 1,
  niter = 2, ncores = 2, se.rule.nct = 3
)
sce_sub <- consensusPart(sce_sub)
```

estDisp *Estimate overdispersion parameter of a beta-binomial*

Description

Estimate overdispersion parameter of a beta-binomial

Usage

```
estDisp(sce, genecluster, type = c("plot", "values"))
```

Arguments

sce	SingleCellExperiment with a1 matrix and counts
genecluster	the gene cluster for which to estimate the over-dispersion parameter. Default is the cluster with the most cells
type	whether to output the over-dispersion estimates as a plot or a value

Value

A ggplot object of the dispersion estimates over the mean, or a data.frame of the mean and dispersion estimates (theta)

Examples

```
sce <- makeSimulatedData()
sce <- preprocess(sce)
sce <- geneCluster(sce, G = seq_len(4))
estDisp(sce, genecluster = 1)
```

---

extractResult *Extract results from an airpart analysis*

Description

results extracts a result table from an airpart analysis giving posterior allelic ratio estimates, s values, false sign rate(fsr), upper confidence interval and lower confidence interval.

Usage

```
extractResult(sce, estimates = c("ar", "svalue", "fsr", "lower", "upper"))
```

Arguments

sce	SingleCellExperiment
estimates	the estimates want to be extracted. Default is allelic ratio estimates, can be "svalue", "fsr", "lower"(credible interval) and "upper"(credible interval)

Value

a DataFrame of estimates

Examples

```
sce <- makeSimulatedData()
sce <- preprocess(sce)
sce <- geneCluster(sce, G = 1:4)
sce_sub <- wilcoxExt(sce, genecluster = 1)
sce_sub <- allelicRatio(sce_sub)
ar <- extractResult(sce_sub)
ar
```

`featureQC` *Quality control on features*

Description

Quality control on features

Usage

```
featureQC(sce, spike, detection_limit = 1, threshold = 0.25, sd = 0.03, pc = 2)
```

Arguments

`sce`	SingleCellExperiment with `counts` and `ratio`
`spike`	the character name of spike genes. The default is `Ercc`
`detection_limit`	Numeric scalar providing the value above which observations are deemed to be expressed.
`threshold`	A numeric scalar specifying the threshold above which percentage of cells expressed within each cell type. Default is 0.25
`sd`	A numeric scalar specifying the cell type weighted allelic ratio mean standard deviation threshold above which are interested features with highly variation. Default is 0.03
`pc`	pseudocount in the `preprocess` step

Value

A DataFrame of QC statistics includes

- filter_celltype indicate whether genes expressed in more than `threshold` cells for all cell types
- sd read counts standard deviation for each feature
- filter_sd indicate whether gene standard deviation exceed `sd`
- filter_spike indicate no spike genes

Examples

```
sce <- makeSimulatedData()
sce <- preprocess(sce)
featureQCmetric <- featureQC(sce)
keep_feature <- (featureQCmetric$filter_celltype &
  featureQCmetric$filter_sd &
  featureQCmetric$filter_spike)
sce <- sce[keep_feature, ]

# or manually setting threshold
featureQCmetric <- featureQC(sce,
  spike = "Ercc",
  threshold = 0.25, sd = 0.03, pc = 2
)
keep_feature <- (featureQCmetric$filter_celltype &
  featureQCmetric$sd > 0.02)
```

fusedLasso *Generalized fused lasso to partition cell types by allelic imbalance*

Description

Fits generalized fused lasso with either binomial(link="logit") or Gaussian likelihood, leveraging functions from the smurf package.

Usage

```
fusedLasso(
  sce,
  formula,
  model = c("binomial", "gaussian"),
  genecluster,
  niter = 1,
  pen.weights,
  lambda = "cv1se.dev",
  k = 5,
  adj.matrix,
  lambda.length = 25L,
  se.rule.nct = 8,
  se.rule.mult = 0.5,
  ...
)
```

Arguments

sce	A SingleCellExperiment containing assays ("ratio", "counts") and colData "x"
formula	A formula object which will typically involve a fused lasso penalty: default is just using cell-type 'x': ratio ~ p(x, pen="gflasso"). Other possibilities would be to use the Graph-Guided Fused Lasso penalty, or add covariates want to be adjusted for, which can include a gene-level baseline 'gene' ratio ~ p(x, pen = "ggflasso") + gene + batch See glmsmurf for more details
model	Either "binomial" or "gaussian" used to fit the generalized fused lasso
genecluster	which gene cluster to run the fused lasso on. Usually one first identifies an interesting gene cluster pattern by summaryAllelicRatio
niter	number of iterations to run; recommended to run 5 times if allelic ratio differences across cell types are within [0.05, 0.1]
pen.weights	argument as described in glmsmurf
lambda	argument as described in glmsmurf. Default lambda is determined by "cv1se.dev" (cross-validation within 1 standard error rule(SE); deviance)
k	number of cross-validation folds
adj.matrix	argument as described in glmsmurf
lambda.length	argument as described in glmsmurf

`se.rule.nct`	the number of cell types to trigger a different SE-based rule than 1 SE (to prioritize larger models, less fusing, good for detecting smaller, e.g. 0.05, allelic ratio differences). When the number of cell types is less than or equal to this value, `se.rule.mult` SE rule is used. When the number of cell types is larger than this value, the standard 1 SE rule is used.
`se.rule.mult`	the multiplier of the SE in determining the lambda: the chosen lambda is within `se.rule.mult` x SE of the minimum deviance. Small values will prioritize larger models, less fusing. Only used when number of cell types is equal to or less than `se.rule.nct`
`...`	additional arguments passed to `glmsmurf`

Details

Usually, we used a Generalized Fused Lasso penalty for the cell states in order to regularize all possible coefficient differences. Another possibility would be to use the Graph-Guided Fused Lasso penalty to only regularize the differences of coefficients of neighboring cell states.

When using a Graph-Guided Fused Lasso penalty, the adjacency matrix corresponding to the graph needs to be provided. The elements of this matrix are zero when two levels are not connected, and one when they are adjacent.

See the package vignette for more details and a complete description of a use case.

Value

A SummarizedExperiment with attached metadata and colData: a matrix grouping factor partition and the penalized parameter lambda are returned in metadata `"partition"` and `"lambda"`. Partition and logistic group allelic estimates are stored in colData: `"part"` and `"coef"`.

References

This function leverages the glmsmurf function from the smurf package. For more details see the following manuscript:

Devriendt S, Antonio K, Reynkens T, et al. Sparse regression with multi-type regularized feature modeling[J]. Insurance: Mathematics and Economics, 2021, 96: 248-261.

See Also

`glmsmurf`, `glmsmurf.control`, `p`, `glm`

Examples

```
library(S4Vectors)
library(smurf)
sce <- makeSimulatedData()
sce <- preprocess(sce)
sce <- geneCluster(sce, G = seq_len(4))
f <- ratio ~ p(x, pen = "gflasso") # formula for the GFL
sce_sub <- fusedLasso(sce,
  formula = f, model = "binomial", genecluster = 1, ncores = 1)
metadata(sce_sub)$partition
metadata(sce_sub)$lambda

# can add covariates or `gene` to the formula
f2 <- ratio ~ p(x, pen = "gflasso") + gene
```

```
sce_sub <- fusedLasso(sce[1:5,],
  formula = f2, model = "binomial",
  genecluster = 1, ncores = 1)

# Suppose we have 4 cell states, if we only want neibouring cell states
# to be grouped together with other cell states. Note here the names of
# the cell states should be given as row and column names.
nct <- nlevels(sce$x)
adjmatrix <- makeOffByOneAdjMat(nct)
colnames(adjmatrix) <- rownames(adjmatrix) <- levels(sce$x)
f <- ratio ~ p(x, pen = "ggflasso") # use graph-guided fused lasso
sce_sub <- fusedLasso(sce,
  formula = f, model = "binomial", genecluster = 1,
  lambda = 0.5, ncores = 1,
  adj.matrix = list(x = adjmatrix)
)
metadata(sce_sub)$partition
```

`geneCluster` *Gene clustering based on allelic ratio matrix with pseudo-count*

Description

Gene clustering based on allelic ratio matrix with pseudo-count

Usage

```
geneCluster(
  sce,
  G,
  method = c("GMM", "hierarchical"),
  minClusterSize = 3,
  plot = TRUE,
  ...
)
```

Arguments

`sce`	SingleCellExperiment containing assays `"ratio_pseudo"` and colData factor `"x"`
`G`	An integer vector specifying the numbers of clusters for which the BIC is to be calculated. The default is G=c(8, 12, 16, 20, 24).
`method`	the method to do gene clustering. The default is the Gaussian Mixture Modeling which is likely to be more accurate. `"hierarchical"` represents automatic hierarchical clustering which is faster to compute.
`minClusterSize`	Minimum cluster size of `"hierarchical"` method.
`plot`	logical, whether to make a PCA plot
`...`	Catches unused arguments in indirect or list calls via do.call as described in `Mclust`

Value

gene cluster IDs are stored in the rowData column `cluster` and a table of gene cluster is returned in metadata `geneCluster`

References

This function leverages Mclust from the mclust package, or hclust.

For mclust see: Luca Scrucca and Michael Fop and T. Brendan Murphy, Adrian E. Raftery "mclust 5: clustering, classification and density estimation using Gaussian finite mixture models" 2016. The R Journal. doi: 10.32614/RJ-2016-021

See Also

`Mclust`

Examples

```
sce <- makeSimulatedData()
sce <- preprocess(sce)
sce <- geneCluster(sce, G = seq_len(4))
```

---

makeForest *Plot allelic ratio result as forest*

Description

Draw a forest plot to visualized cell type specific allelic ratio estimator and confidence interval. It is based on the **forestplot**-package's `forestplot` function.

Usage

```
makeForest(
  sce,
  genepoi,
  ctpoi = seq_len(nlevels(sce$x)),
  showtext = FALSE,
  xticks,
  boxsize = 0.25,
  xlab = "Allelic Ratio",
  col,
  grid = structure(seq(0.1, 0.9, 0.1), gp = gpar(lty = 2, col = "#CCCCFF")),
  ...
)
```

Arguments

`sce`	A SingleCellExperiment containing colData allelic ratio estimator in the third column and last two column is the confidence interval.
`genepoi`	the gene position index or gene name vector that want to be plotted. Ordered by increased cell type svalue. Default is the top 40 genes that has minimum svalue in any cell type or all genes if number of genes smaller than 40.

ctpoi	the cell type position index that want to be plotted.
showtext	indicate whether show the svalue information along the forestplot.
xticks	argument as described in forestplot
boxsize	Override the default box size based on precision
xlab	x-axis label. Default is "Allelic Ratio"
col	Set the colors for all the elements. See fpColors for details
grid	If you want a discrete gray dashed grid at the level of the ticks you can set this parameter to TRUE. If you set the parameter to a vector of values lines will be drawn at the corresponding positions. If you want to specify the gpar of the lines then either directly pass a gpar object or set the gp attribute e.g. attr(line_vector, "gp") <- gpar(lty=2, col = "red")
...	Passsed on the other argument in forestplot.

Value

generates a forest plot

See Also

forestplot, fpColors, fpShapesGp, fpLegend

Examples

```
sce <- makeSimulatedData()
sce <- preprocess(sce)
sce <- geneCluster(sce, G = 1:4)
sce_sub <- wilcoxExt(sce, genecluster = 1)
sce_sub <- allelicRatio(sce_sub)
makeForest(sce_sub, showtext = TRUE)

# if want to change some properties, like ticks position
library(forestplot)
xticks <- seq(from = 0, to = 1, by = 0.25)
xtlab <- rep(c(TRUE, FALSE), length.out = length(xticks))
attr(xticks, "labels") <- xtlab
genepoi <- paste0("gene", seq_len(5))
ctpoi <- c(1, 3)
makeForest(sce_sub, genepoi, ctpoi,
  xticks = xticks,
  col = fpColors(box = c("blue", "red", "black", "darkgreen"))
)
```

---

makeHeatmap *Plot allelic ratio as heatmap*

Description

Plot allelic ratio as heatmap

Usage

```
makeHeatmap(
  sce,
  assay = c("ratio_pseudo", "ratio", "counts"),
  genecluster = NULL,
  show_row_names = FALSE,
  order_by_group = TRUE,
  ...
)
```

Arguments

sce	SingleCellExperiment
assay	the assay to be plotted. Choices are "ratio_pseudo" which is the default, "ratio", "counts".
genecluster	an integer indicates which gene cluster heatmap want to be returned.
show_row_names	show row names or not
order_by_group	indicate whether order by group or order by cell types
...	Passsed on the other argument in Heatmap.

Value

generates a heatmap

Examples

```
set.seed(2021)
sce <- makeSimulatedData(p.vec = c(0.3, 0.5, 0.5, 0.3), ncl = 1)
sce <- preprocess(sce)
# display allelic ratio pattern in whole dataset
makeHeatmap(sce)

sce <- geneCluster(sce, G = seq_len(4), plot = FALSE)
sce_sub <- wilcoxExt(sce, genecluster = 1)
# display specific gene cluster partition result
makeHeatmap(sce_sub)
# display by cell type orders
makeHeatmap(sce_sub, order_by_group = FALSE)
```

---

makeOffByOneAdjMat *Generating adjancy matrix for neighboring cell states.*

Description

To use the Graph-Guided Fused Lasso penalty to only regularize the differences of coefficients of neighboring areas, suitable for time/spatial analysis. The adjacency matrix corresponding to the graph needs to be provided. The elements of this matrix are zero when two levels are not connected, and one when they are adjacent.

Usage

```
makeOffByOneAdjMat(nct)
```

Arguments

nct	the number of cell types/states

Details

If manually input the adjacency matrix, this matrix has to be symmetric and the names of the cell states should be given as row and column names.

Examples

```
sce <- makeSimulatedData()
nct <- nlevels(sce$x)
adjmatrix <- makeOffByOneAdjMat(nct)
colnames(adjmatrix) <- rownames(adjmatrix) <- levels(sce$x)
```

---

makeSimulatedData *Make simulated data for airpart*

Description

Make simulated data for airpart

Usage

```
makeSimulatedData(
  mu1 = 2,
  mu2 = 10,
  nct = 4,
  n = 30,
  ngenecl = 50,
  theta = 20,
  ncl = 3,
  p.vec = rep(c(0.2, 0.8, 0.5, 0.5, 0.7, 0.9), each = 2),
  totalClusters = FALSE
)
```

Arguments

mu1	low count (typical of "noisy" ratio estimates)
mu2	high count
nct	number of cell types
n	number of cells per cell type
ngenecl	number of genes per cluster
theta	overdispersion parameter (higher is closer to binomial)
ncl	number of gene cluster
p.vec	the allelic ratio vector which follows gene cluster order. (length is nct * ncl)
totalClusters	logical, whether cell types should cluster by total count

Value

SingleCellExperiment with the following elements as assays

- a1 allelic count matrix for the numerator/effect allele
- a2 allelic count matrix for the denominator/non-effect allele
- true.ratio a matrix of the true probabilities (allelic ratios) for the cell types

Also x in the colData is a vector of annotated cell types in the same order as cells in count matrix

Examples

```
library(SummarizedExperiment)
sce <- makeSimulatedData()
assayNames(sce)
```

---

makeStep *Plot group partition and posterior allelic ratio estimates by step*

Description

Plot group partition and posterior allelic ratio estimates by step

Usage

```
makeStep(sce, xlab = "cell type")
```

Arguments

sce	SingleCellExperiment
xlab	the x axis name.

Value

a ggplot2 object.

Examples

```
sce <- makeSimulatedData()
sce <- preprocess(sce)
sce <- geneCluster(sce, G = 1:4)
sce_sub <- wilcoxExt(sce, genecluster = 1)
sce_sub <- allelicRatio(sce_sub)
makeStep(sce_sub)
```

makeViolin *Posterior mean allelic ratio estimates in violin plots*

Description

Posterior mean allelic ratio estimates in violin plots

Usage

```
makeViolin(sce, xlab = "cell type", ylim = c(0, 1))
```

Arguments

sce	SingleCellExperiment
xlab	the x axis name.
ylim	the y axis range

Value

a ggplot2 object, n represents number of cells in that cell type.

Examples

```
sce <- makeSimulatedData()
sce <- preprocess(sce)
sce <- geneCluster(sce, G = 1:4)
sce_sub <- wilcoxExt(sce, genecluster = 1)
sce_sub <- allelicRatio(sce_sub)
makeViolin(sce_sub)
```

preprocess *Preprocess the SingleCellExperiment*

Description

Preprocess the SingleCellExperiment

Usage

```
preprocess(sce, pc = 2)
```

Arguments

sce	SingleCellExperiment with a1 (effect allele) and a2 (non-effect allele). The allelic ratio will be calculated as a1 / (a1 + a2).
pc	pseudocount for calculating the smoothed ratio

Value

SingleCellExperiment with total count, allelic ratio = a1/(a1 + a2), and pseud-ocount-smoothed ratio

Examples

```
library(SummarizedExperiment)
sce <- makeSimulatedData()
sce <- preprocess(sce)
assayNames(sce)
```

`summaryAllelicRatio` *Allelic ratio summary*

Description

Oroduce allelic ratio summaries for each gene cluster

Usage

```
summaryAllelicRatio(sce, genecluster)
```

Arguments

`sce`	SingleCellExperiment
`genecluster`	an optional vector of gene cluster IDs. if nothing is given, all cluster's summaries will be calculated

Value

a list of gene cluster summary tables containing:

- weighted.mean weighted mean of allelic ratio for the cell types
- mean mean allelic ratio for the cell types
- var variance of allelic ratio for the cell types

Examples

```
library(S4Vectors)
sce <- makeSimulatedData()
sce <- preprocess(sce)
sce <- geneCluster(sce, G = 1:4)
summary <- summaryAllelicRatio(sce, genecluster = c(1, 3))
summary
```

wilcoxExt *Extension on Pairwise Mann Whitney Wilcoxon Test for partitioning*

Description

Extends the Pairwise Mann Whitney Wilcoxon Test by combining hierarchical clustering for partition.

Usage

```
wilcoxExt(
  sce,
  genecluster,
  threshold,
  adj.matrix,
  p.adjust.method = "none",
  ncores = NULL,
  ...
)
```

Arguments

sce	A SingleCellExperiment containing assays ("ratio", "counts") and colData "x"
genecluster	which gene cluster result want to be returned. Usually identified interesting gene cluster pattern by summaryAllelicRatio
threshold	a vector with candidate thresholds for raw p-value cut-off. Default is 10^seq(from=-2,to=-0.4,by=0.2). For details please see vignette
adj.matrix	an adjacency matrix with 1 indicates cell states allowed to be grouped together, 0 otherwise.
p.adjust.method	method for adjusting p-values (see p.adjust). Can be abbreviated
ncores	A cluster object created by makeCluster. Or an integer to indicate number of child-processes (integer values are ignored on Windows) for parallel evaluations
...	additional arguments to pass to wilcox.test.

Value

A matrix grouping factor partition and the significant cut-off threshold are returned in metadata "partition" and "threshold". Partation also stored in colData"part". Note we recommend the returned "threshold" is not at the ends of input "threshold".

Examples

```
library(S4Vectors)
sce <- makeSimulatedData()
sce <- preprocess(sce)
sce <- geneCluster(sce, G = seq_len(4))
sce_sub <- wilcoxExt(sce, genecluster = 1)
metadata(sce_sub)$partition
```

```
metadata(sce_sub)$threshold

# Suppose we have 4 cell states, if we don't want cell state 1
# to be grouped together with other cell states
adj.matrix <- 1 - diag(4)
colnames(adj.matrix) <- rownames(adj.matrix) <- levels(sce$x)
adj.matrix[1, c(2, 3, 4)] <- 0
adj.matrix[c(2, 3, 4), 1] <- 0
thrs <- 10^seq(from = -2, to = -0.4, by = 0.1)
sce_sub <- wilcoxExt(sce,
  genecluster = 1, threshold = thrs,
  adj.matrix = adj.matrix
)
metadata(sce_sub)$partition
```

Index

allelicRatio, 2
apeglm, *3*

cellQC, 3
consensusPart, 5

estDisp, 5
extractResult, 6

featureQC, 7
forestplot, *12*
formula, *3*, *8*
fpColors, *12*
fpLegend, *12*
fpShapesGp, *12*
fusedLasso, *3*, 8

geneCluster, 10
glm, *9*
glmsmurf, *8*, *9*
glmsmurf.control, *9*
gpar, *12*

Heatmap, *13*

makeCluster, *18*
makeForest, 11
makeHeatmap, 12
makeOffByOneAdjMat, 13
makeSimulatedData, 14
makeStep, 15
makeViolin, 16
Mclust, *10*, *11*

ns, *3*

p, *9*
p.adjust, *18*
preprocess, 16

summaryAllelicRatio, *8*, 17, *18*

wilcox.test, *18*
wilcoxExt, 18