Package ‘muscat’

April 15, 2020

Title Multi-sample multi-group scRNA-seq data analysis tools

Description `muscat` provides various methods and visualization tools for DS analysis in multi-sample, multi-group, multi-(cell-)subpopulation scRNA-seq data, including cell-level mixed models and methods based on aggregated “pseudobulk” data, as well as a flexible simulation platform that mimics both single and multi-sample scRNA-seq data.

Type Package

Version 1.0.1

Depends R (>= 3.6), scater (>= 1.13)

Imports BiocParallel, blme, ComplexHeatmap, data.table, DESeq2, doParallel, dplyr, edgeR, ggplot2, glmmTMB, grDevices, grid, limma, lmerTest, lme4, magrittr, Matrix, matrixStats, methods, parallel, progress, purrr, reshape2, scales, sctransform, stats, SingleCellExperiment, SummarizedExperiment, S4Vectors, tibble, variancePartition, viridis

Suggests BiocStyle, cowplot, ExperimentHub, knitr, RColorBrewer, rmarkdown, testthat, UpSetR

biocViews DifferentialExpression, Sequencing, SingleCell, Software, StatisticalMethod, Visualization

License GPL (>= 2)

VignetteBuilder knitr

RoxygenNote 7.1.0

Encoding UTF-8

LazyData TRUE

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

git_branch RELEASE_3_10

git_last_commit 9202144

git_last_commit_date 2020-03-31

Date/Publication 2020-04-14

Author Helena L. Crowell [aut, cre],
Charlotte Soneson [aut],
Pierre-Luc Germain [aut],
Mark D. Robinson [aut, fnd]

Maintainer Helena L. Crowell <helena.crowell@uzh.ch>

R topics documented:

aggregateData . . . 2
calcExprFreqs . . . 3
data . . . 4
mmDS . . . 6
pbDS . . . 9
pbHeatmap . . . 10
pbMDS . . . 12
prepSCE . . . 12
prepSim . . . 13
resDS . . . 15
simData . . . 17

Index **19**

---

aggregateData *Aggregation of single-cell to pseudobulk data*

Description

...

Usage

```
aggregateData(
  x,
  assay = NULL,
  by = c("cluster_id", "sample_id"),
  fun = c("sum", "mean", "median"),
  scale = FALSE
)
```

Arguments

x	a SingleCellExperiment.
assay	character string specifying the assay slot to use as input data. Defaults to the 1st available (assayNames(x)[1]).
by	character vector specifying which colData(x) columns to summarize by (at most 2!).
fun	a character string. Specifies the function to use as summary statistic.
scale	logical. Should pseudo-bulks be scaled with the effective library size & multiplied by 1M?

Value

a SingleCellExperiment.

- If length(by) == 2, each sheet (assay) contains pseudobulks for each of by[1], e.g., for each cluster when by = "cluster_id". Rows correspond to genes, columns to by[2], e.g., samples when by = "sample_id".

- If `length(by) == 1`, the returned SCE will contain only a single assay with rows = genes and colums = by.

Aggregation parameters (`assay,by,fun,scaled`) are stored in `metadata()$agg_pars`, and the number of cells that were aggregated are accessible in `metadata()$n_cells`.

Author(s)

Helena L Crowell & Mark D Robinson

References

Crowell, HL, Soneson, C, Germain, P-L, Calini, D, Collin, L, Raposo, C, Malhotra, D & Robinson, MD: On the discovery of population-specific state transitions from multi-sample multi-condition single-cell RNA sequencing data. *bioRxiv* **713412** (2018). doi: https://doi.org/10.1101/713412

Examples

```
data(sce)
library(SingleCellExperiment)

# pseudobulk counts by cluster-sample
pb <- aggregateData(sce)

assayNames(sce)  # one sheet per cluster
head(assay(sce)) # n_genes x n_samples

# scaled CPM
assays(sce)$cpm <- edgeR::cpm(assay(sce))
pb <- aggregateData(sce, assay = "cpm", scale = TRUE)
head(assay(pb))

# aggregate by cluster only
pb <- aggregateData(sce, by = "cluster_id")
length(assays(pb)) # single assay
head(assay(pb))    # n_genes x n_clusters
```

`calcExprFreqs` *calcExprFreqs*

Description

Calculates gene expression frequencies

Usage

```
calcExprFreqs(x, assay = "counts", th = 0)
```

Arguments

`x`	a `SingleCellExperiment`.
`assay`	a character string specifying which assay to use.
`th`	numeric threshold value above which a gene should be considered to be expressed.

Details

`calcExprFreq` computes, for each sample and group (in each cluster), the fraction of cells that express a given gene. Here, a gene is considered to be expressed when the specified measurement value (`assay`) lies above the specified threshold value (`th`).

Value

a `SingleCellExperiment` containing, for each cluster, an assay of dimensions #genes x #samples giving the fraction of cells that express each gene in each sample. If `colData(x)` contains a `"group_id"` column, the fraction of expressing cells in each each group will be included as well.

Author(s)

Helena L Crowell & Mark D Robinson

Examples

```
data(sce)
library(SingleCellExperiment)

frq <- calcExprFreqs(sce)

# one assay per cluster
assayNames(frq)

# expression frequencies by
# sample & group; 1st cluster:
head(assay(frq))
```

`data` *Example datasets*

Description

A `SingleCellExperiment` containing 10x droplet-based scRNA-seq PBCM data from 8 Lupus patients befor and after 6h-treatment with INF-beta (16 samples in total).

The original data has been filtered to

- remove unassigned cells & cell multiplets
- retain only 4 out of 8 samples per experimental group
- retain only 5 out of 8 subpopulations (clusters)
- retain genes with a count > 1 in > 50 cells

- retain cells with > 200 detected genes
- retain at most 100 cells per cluster-sample instance

Assay `logcounts` corresponds to log-normalized values obtained from `normalize` with default parameters.

The original measurement data, as well as gene and cell metadata is available through the NCBI GEO accession number GSE96583; code to reproduce this example dataset from the original data is provided in the examples section.

Value

a `SingleCellExperiment`.

Author(s)

Helena L Crowell

References

Kang et al. (2018). Multiplexed droplet single-cell RNA-sequencing using natural genetic variation. *Nature Biotechnology*, **36**(1): 89-94. DOI: 10.1038/nbt.4042.

Examples

```
## Not run:
# set random seed for cell sampling
set.seed(2929)

# load data
library(ExperimentHub)
eh <- ExperimentHub()
sce <- eh[["EH2259"]]

# drop unassigned cells & multiplets
sce <- sce[, !is.na(sce$cell)]
sce <- sce[, sce$multiplets == "singlet"]

# keep 4 samples per group
sce$id <- paste0(sce$stim, sce$ind)
inds <- sample(sce$ind, 4)
ids <- paste0(levels(sce$stim), rep(inds, each = 2))
sce <- sce[, sce$id %in% ids]

# keep 5 clusters
kids <- c("B cells", "CD4 T cells", "CD8 T cells",
    "CD14+ Monocytes", "FCGR3A+ Monocytes")
sce <- sce[, sce$cell %in% kids]
sce$cell <- droplevels(sce$cell)

# basic filtering on  genes & cells
gs <- rowSums(counts(sce) > 1) > 50
cs <- colSums(counts(sce) > 0) > 200
sce <- sce[gs, cs]

# sample max. 100 cells per cluster-sample
cs_by_ks <- split(colnames(sce), list(sce$cell, sce$id))
```

```
cs <- sapply(cs_by_ks, function(u)
    sample(u, min(length(u), 100)))
sce <- sce[, unlist(cs)]

# compute logcounts
library(scater)
sce <- computeLibraryFactors(sce)
sce <- logNormCounts(sce)

# re-format for 'muscat'
sce <- prepSCE(sce,
    cluster_id = "cell",
    sample_id = "id",
    group_id = "stim",
    drop = TRUE)

## End(Not run)
```

---

mmDS *DS analysis using mixed-models (MM)*

Description

Performs cluster-wise DE analysis by fitting cell-level models.

Usage

```
mmDS(
  x,
  coef = NULL,
  covs = NULL,
  method = c("dream", "vst", "poisson", "nbinom", "hybrid"),
  n_cells = 10,
  n_samples = 2,
  min_count = 1,
  min_cells = 20,
  n_threads = 8,
  verbose = TRUE,
  dup_corr = FALSE,
  trended = FALSE,
  vst = c("sctransform", "DESeq2"),
  bayesian = FALSE,
  blind = TRUE,
  REML = TRUE,
  ddf = c("Satterthwaite", "Kenward-Roger", "lme4")
)

.mm_dream(
  x,
  coef = NULL,
  covs = NULL,
```

```
  dup_corr = FALSE,
  trended = FALSE,
  ddf = c("Satterthwaite", "Kenward-Roger"),
  n_threads = 1,
  verbose = FALSE
)

.mm_vst(
  x,
  vst = c("sctransform", "DESeq2"),
  coef = NULL,
  covs = NULL,
  bayesian = FALSE,
  blind = TRUE,
  REML = TRUE,
  ddf = c("Satterthwaite", "Kenward-Roger", "lme4"),
  n_threads = 1,
  verbose = FALSE
)
```

Arguments

`x`	a `SingleCellExperiment`.
`coef`	character specifying the coefficient to test. If NULL (default), will test the last level of `"group_id"`.
`covs`	character vector of `colData(x)` column names to use as covariates.
`method`	a character string. Either `"dream"` (default, lme4 with voom-weights), `"vst"` (variance-stabilizing transformation), `"poisson"` (poisson GLM-MM), `"nbinom"` (negative binomial GLM-MM), `"hybrid"` (combination of pseudobulk and poisson methods) or a function accepting the same arguments.
`n_cells`	number of cells per cluster-sample required to consider a sample for testing.
`n_samples`	number of samples per group required to consider a cluster for testing.
`min_count`	numeric. For a gene to be tested in a given cluster, at least `min_cells` must have a count >= `min_count`.
`min_cells`	number (or fraction, if < 1) of cells with a count > `min_count` required for a gene to be tested in a given cluster.
`n_threads`	number of threads to use.
`verbose`	logical specifying whether messages on progress and a progress bar should be displayed.
`dup_corr`	logical; whether to use `duplicateCorrelation`.
`trended`	logical; whether to use expression-dependent variance priors in `eBayes`.
`vst`	method to use as variance-stabilizing transformations. `"sctransform"` for `vst`; `"DESeq2"` for `varianceStabilizingTransformation`.
`bayesian`	logical; whether to use bayesian mixed models.
`blind`	logical; whether to ignore experimental design for the vst.
`REML`	logical; whether to maximize REML instead of log-likelihood.
`ddf`	character string specifying the method for estimating the effective degrees of freedom. For `method = "dream"`, either `"Satterthwaite"` (faster) or `"Kenward-Roger"` (more accurate); see `?variancePartition::dream` for details. For `method = "vst"`, method `"lme4"` is also valid; see `contest.lmerModLmerTest`.

Details

.mm_dream and .mm_vst expect cells from a single cluster, and do not perform filtering or handle incorrect parameters well. Meant to be called by mmDS with method = c("dream","vst") and vst = c("sctransform","DESeq2") to be applied across all clusters.

method = "dream" variancePartition's voom-lme4-implementation of mixed models for RNA-seq data; function dream.

method = "vst" vst = "sctransform" lmer or blmer mixed models on vst transformed counts.

vst = "DESeq2" varianceStabilizingTransformation followed by lme4 mixed models.

Value

a data.frame

Functions

- .mm_dream: see details.
- .mm_vst: see details.

Author(s)

Pierre-Luc Germain & Helena L Crowell

References

Crowell, HL, Soneson, C, Germain, P-L, Calini, D, Collin, L, Raposo, C, Malhotra, D & Robinson, MD: On the discovery of population-specific state transitions from multi-sample multi-condition single-cell RNA sequencing data. *bioRxiv* **713412** (2018). doi: https://doi.org/10.1101/713412

Examples

```
data(sce)
# subset "B cells" cluster
sce <- sce[, sce$cluster_id == "B cells"]
sce$cluster_id <- droplevels(sce$cluster_id)

# downsample to 100 genes
cs_by_s <- split(colnames(sce), sce$sample_id)
gs <- sample(nrow(sce), 100)
sce <- sce[gs, ]

res <- mmDS(sce, method = "dream",
    n_threads = 1, verbose = FALSE)
head(res$`B cells`)
```

pbDS *pseudobulk DS analysis*

Description

pbDS tests for DS after aggregating single-cell measurements to pseudobulk data, by applying bulk RNA-seq DE methods, such as edgeR, DESeq2 and limma.

Usage

```
pbDS(
  pb,
  method = c("edgeR", "DESeq2", "limma-trend", "limma-voom"),
  design = NULL,
  coef = NULL,
  contrast = NULL,
  min_cells = 10,
  verbose = TRUE
)
```

Arguments

pb	a SingleCellExperiment containing pseudobulks as returned by aggregateData.
method	a character string.
design	For methods "edegR" and "limma", a design matrix with row & column names(!) created with model.matrix; For "DESeq2", a formula with variables in colData(pb). Defaults to ~ group_id or the corresponding model.matrix.
coef	passed to glmQLFTest, contrasts.fit, results for method = "edgeR", "limma-x", "DESeq2", respectively.
contrast	a matrix of contrasts to test for created with makeContrasts.
min_cells	a numeric. Specifies the minimum number of cells in a given cluster-sample required to consider the sample for differential testing.
verbose	logical. Should information on progress be reported?

Value

a list containing

- a data.frame with differential testing results,
- a DGEList object of length nb.-clusters, and
- the design matrix, and contrast or coef used.

Author(s)

Helena L Crowell & Mark D Robinson

References

Crowell, HL, Soneson, C, Germain, P-L, Calini, D, Collin, L, Raposo, C, Malhotra, D & Robinson, MD: On the discovery of population-specific state transitions from multi-sample multi-condition single-cell RNA sequencing data. *bioRxiv* **713412** (2018). doi: https://doi.org/10.1101/713412

Examples

```
# simulate 5 clusters, 20% of DE genes
data(sce)

# compute pseudobulk sum-counts & run DS analysis
pb <- aggregateData(sce)
res <- pbDS(pb, method = "limma-trend")

names(res)
names(res$table)
head(res$table$`stim-ctrl`$`B cells`)

# count nb. of DE genes by cluster
vapply(res$table$`stim-ctrl`, function(u)
  sum(u$p_adj.loc < 0.05), numeric(1))

# get top 5 hits for ea. cluster w/ abs(logFC) > 1
library(dplyr)
lapply(res$table$`stim-ctrl`, function(u)
  filter(u, abs(logFC) > 1) %>%
    arrange(p_adj.loc) %>%
    slice(seq_len(5)))
```

---

pbHeatmap — *Heatmap of cluster-sample pseudobulks*

Description

...

Usage

```
pbHeatmap(
  x,
  y,
  k = NULL,
  g = NULL,
  c = NULL,
  top_n = 20,
  fdr = 0.05,
  lfc = 1,
  sort_by = "p_adj.loc",
  decreasing = FALSE,
  assay = "logcounts",
```

```
    fun = mean,
    normalize = TRUE,
    col = viridis(10),
    row_anno = TRUE,
    col_anno = TRUE
  )
```

Arguments

- `x` — a `SingleCellExperiment`.
- `y` — a list of DS analysis results as returned by `pbDS` or `mmDS`.
- `k` — character vector; specifies which cluster ID(s) to retain. Defaults to `levels(x$cluster_id)`.
- `g` — character vector; specifies which genes to retain. Defaults to considering all genes.
- `c` — character string; specifies which contrast/coefficient to retain. Defaults to `names(y$table)[1]`.
- `top_n` — single numeric; number of genes to retain per cluster.
- `fdr, lfc` — single numeric; FDR and logFC cutoffs to filter results by. The specified FDR threshold is applied to `p_adj.loc` values.
- `sort_by` — character string specifying a numeric results table column to sort by.
- `decreasing` — logical; whether to sort in decreasing order of `sort_by`.
- `assay` — character string; specifies which assay to use; should be one of `assayNames(x)`.
- `fun` — function to use as summary statistic, e.g., mean, median, sum (depending on the input assay).
- `normalize` — logical; whether to apply a z-normalization to each row (gene) of the cluster-sample pseudobulk data.
- `col` — character vector of colors or color mapping function generated with `colorRamp2`. Passed to argument `col` in `Heatmap` (see `?ComplexHeatmap::Heatmap` for details).
- `row_anno, col_anno` — logical; whether to render annotations of cluster and group IDs, respectively.

Value

a `HeatmapList-class` object.

Author(s)

Helena L Crowell

Examples

```
data(sce)

# compute pseudobulks & run DS analysis
pb <- aggregateData(sce)
res <- pbDS(pb)

# cluster-sample expression means
pbHeatmap(sce, res)
```

```
# include only a single cluster
pbHeatmap(sce, res, k = "B cells")

# plot specific gene across all clusters
pbHeatmap(sce, res, g = "ISG20")
```

`pbMDS` *Pseudobulk-level MDS plot*

Description

Renders a multidimensional scaling (MDS) where each point represents a cluster-sample instance; with points colored by cluster ID and shaped by group ID.

Usage

```
pbMDS(x)
```

Arguments

`x`	a `SingleCellExperiment` containing cluster-sample pseudobulks as returned by `aggregateData` with argument by = `c("cluster_id","sample_id")`.

Value

a `ggplot` object.

Author(s)

Helena L Crowell & Mark D Robinson

Examples

```
data(sce)
pb <- aggregateData(sce)
pbMDS(pb)
```

`prepSCE` *Prepare SCE for DS analysis*

Description

...

Usage

```
prepSCE(x, cluster_id, sample_id, group_id, drop = FALSE)
```

Arguments

`x` a `SingleCellExperiment`.

`cluster_id, sample_id, group_id`
: character strings specifying the `colData(x)` columns containing cluster assignments, unique sample identifiers, and group IDs (e.g., treatment).

`drop` logical. Specifies whether `colData(x)` columns besides those specified as `cluster_id,sample_id,` should be retained (default `drop = FALSE`) or removed (`drop = TRUE`).

Value

a `SingleCellExperiment`.

Author(s)

Helena L Crowell

Examples

```
# generate random counts
ng <- 50
nc <- 200
counts <- matrix(sample(ng * nc), nrow = ng, ncol = nc)

# generate some cell metadata
gids <- sample(c("groupA", "groupB"), nc, TRUE)
sids <- sample(paste0("sample", seq_len(3)), nc, TRUE)
kids <- sample(paste0("cluster", seq_len(5)), nc, TRUE)
batch <- sample(seq_len(3), nc, TRUE)

# construct SCE
library(SingleCellExperiment)
sce <- SingleCellExperiment(
  assays = list(counts = counts),
  colData = data.frame(group = gids, id = sids, cluster = kids, batch))

# prep. for workflow
sce <- prepSCE(sce,
  group_id = "group",
  sample_id = "id",
  cluster_id = "cluster")

head(colData(sce))
metadata(sce)$experiment_info
```

`prepSim` *SCE preparation for* `simData`

Description

`prepSim` prepares an input SCE for simulation with `muscat`'s `simData` function by

1. basic filtering of genes and cells
2. (optional) filtering of subpopulation-sample instances
3. estimation of cell (library sizes) and gene parameters (dispersions and sample-specific means), respectively.

Usage

```
prepSim(
  x,
  min_count = 1,
  min_cells = 10,
  min_genes = 100,
  min_size = 100,
  group_keep = NULL,
  verbose = TRUE
)
```

Arguments

`x`	a `SingleCellExperiment`.
`min_count, min_cells`	used for filtering of genes; only genes with a count > `min_count` in >= `min_cells` will be retained.
`min_genes`	used for filtering cells; only cells with a count > 0 in >= `min_genes` will be retained.
`min_size`	used for filtering subpopulation-sample combinations; only instances with >= `min_size` cells will be retained. Specifying `min_size = NULL` skips this step.
`group_keep`	character string; if `nlevels(x$group_id) > 1`, specifies which group of samples to keep (see details). The default NULL retains samples from `levels(x$group_id)[1]`; otherwise, if 'colData(x)$group_id' is not specified, all samples will be kept.
`verbose`	logical; should information on progress be reported?

Details

For each gene g, `prepSim` fits a model to estimate sample-specific means β_g^s, for each sample s, and dispersion parameters ϕ_g using `edgeR`'s `estimateDisp` function with default parameters. Thus, the reference count data is modeled as NB distributed:

$$Y_{gc} \sim NB(\mu_{gc}, \phi_g)$$

for gene g and cell c, where the mean $\mu_{gc} = \exp(\beta_g^{s(c)}) \cdot \lambda_c$. Here, $\beta_g^{s(c)}$ is the relative abundance of gene g in sample $s(c)$, λ_c is the library size (total number of counts), and ϕ_g is the dispersion.

Value

a `SingleCellExperiment` containing, for each cell, library size (`colData(x)$offset`) and, for each gene, dispersion and sample-specific mean estimates (`rowData(x)$dispersion` and `$beta.sample_id`, respectively).

Author(s)

Helena L Crowell

References

Crowell, HL, Soneson, C, Germain, P-L, Calini, D, Collin, L, Raposo, C, Malhotra, D & Robinson, MD: On the discovery of population-specific state transitions from multi-sample multi-condition single-cell RNA sequencing data. *bioRxiv* **713412** (2018). doi: https://doi.org/10.1101/713412

Examples

```
data(sce)
library(SingleCellExperiment)

sce2 <- prepSim(sce)

# nb. of genes/cells before vs. after
cbind(before = dim(sce), after = dim(sce2))

head(rowData(sce2)) # gene parameters
head(colData(sce2)) # cell parameters
```

---

resDS — *resDS Formatting of DS analysis results*

Description

resDS provides a simple wrapper to format cluster-level differential testing results into an easily filterable table, and to optionally append gene expression frequencies by cluster-sample & -group, as well as cluster-sample-wise CPM.

Usage

```
resDS(
  x,
  y,
  bind = c("col", "row"),
  frq = FALSE,
  cpm = FALSE,
  digits = 3,
  sep = "__",
  ...
)
```

Arguments

x	a SingleCellExperiment.
y	a list of DS testing results as returned by pbDS or mmDS.
bind	character string specifying the output format (see details).

frq	logical or a pre-computed list of expression frequencies as returned by calcExprFreqs.
cpm	logical specifying whether CPM by cluster-sample should be appendeded to the output result table(s).
digits	integer value specifying the number of significant digits to maintain.
sep	character string to use as separator when constructing new column names.
...	optional arguments passed to calcExprFreqs if frq = TRUE.

Details

When bind = "col", the list of DS testing results at y$table will be merge vertically (by column) into a single table in tidy format with column contrast/coef specifying the comparison.

Otherwise, when bind = "row", an identifier of the respective contrast or coefficient will be appended to the column names, and all tables will be merge horizontally (by row).

Expression frequencies pre-computed with calcExprFreqs may be provided with frq. Alternatively, when frq = TRUE, expression frequencies can be computed directly, and additional arguments may be passed to calcExprFreqs (see examples below).

Value

returns a ‘data.frame‘.

Author(s)

Helena L Crowell & Mark D Robinson

Examples

```
data(sce)

# compute pseudobulks (sum of counts)
pb <- aggregateData(sce, assay = "counts", fun = "sum")

# run DS analysis (edgeR on pseudobulks)
res <- pbDS(pb, method = "edgeR")

head(resDS(sce, res, bind = "row")) # tidy format
head(resDS(sce, res, bind = "col", digits = Inf))

# append CPMs & expression frequencies
head(resDS(sce, res, cpm = TRUE))
head(resDS(sce, res, frq = TRUE))

# pre-computed expression frequencies & append
frq <- calcExprFreqs(sce, assay = "counts", th = 0)
head(resDS(sce, res, frq = frq))
```

simData *simData*

Description

Simulation of complex scRNA-seq data

Usage

```
simData(
  x,
  n_genes = 500,
  n_cells = 300,
  probs = NULL,
  p_dd = diag(6)[1, ],
  p_type = 0,
  lfc = 2,
  rel_lfc = NULL
)
```

Arguments

x	a SingleCellExperiment.
n_genes	# of genes to simulate.
n_cells	# of cells to simulate. Either a single numeric or a range to sample from.
probs	a list of length 3 containing probabilities of a cell belonging to each cluster, sample, and group, respectively. List elements must be NULL (equal probabilities) or numeric values in [0, 1] that sum to 1.
p_dd	numeric vector of length 6. Specifies the probability of a gene being EE, EP, DE, DP, DM, or DB, respectively.
p_type	numeric. Probaility of EE/EP gene being a type-gene. If a gene is of class "type" in a given cluster, a unique mean will be used for that gene in the respective cluster.
lfc	numeric value to use as mean logFC for DE, DP, DM, and DB type of genes.
rel_lfc	numeric vector of relative logFCs for each cluster. Should be of length `nlevels(x$cluster_id)` with `levels(x$cluster_id)` as names. Defaults to factor of 1 for all clusters.

Details

`simData` simulates multiple clusters and samples across 2 experimental conditions from a real scRNA-seq data set.

Value

a SingleCellExperiment containing multiple clusters & samples across 2 groups.

Author(s)

Helena L Crowell

References

Crowell, HL, Soneson, C, Germain, P-L, Calini, D, Collin, L, Raposo, C, Malhotra, D & Robinson, MD: On the discovery of population-specific state transitions from multi-sample multi-condition single-cell RNA sequencing data. *bioRxiv* **713412** (2018). doi: https://doi.org/10.1101/713412

Examples

```
data(sce)
library(SingleCellExperiment)

# prep. SCE for simulation
sce <- prepSim(sce)

# simulate data
(sim <- simData(sce,
  n_genes = 100, n_cells = 10,
  p_dd = c(0.9, 0, 0.1, 0, 0, 0)))

# simulation metadata
head(gi <- metadata(sim)$gene_info)

# should be ~10% DE
table(gi$category)

# unbalanced sample sizes
sim <- simData(sce,
  n_genes = 10, n_cells = 100,
  probs = list(NULL, c(0.25, 0.75), NULL))
table(sim$sample_id)

# one group only
sim <- simData(sce,
  n_genes = 10, n_cells = 100,
  probs = list(NULL, NULL, c(1, 0)))
levels(sim$group_id)
```

Index

.mm_dream (mmDS), 6
.mm_vst (mmDS), 6

aggregateData, 2, 9, 12

calcExprFreqs, 3, 16
colorRamp2, 11
contest.lmerModLmerTest, 7
contrasts.fit, 9

data, 4
DGEList, 9
duplicateCorrelation, 7

eBayes, 7
estimateDisp, 14

glmQLFTest, 9

Heatmap, 11

makeContrasts, 9
mmDS, 6, 11, 15
model.matrix, 9

normalize, 5

pbDS, 9, 11, 15
pbHeatmap, 10
pbMDS, 12
prepSCE, 12
prepSim, 13

resDS, 15
results, 9

sce (data), 4
simData, 13, 14, 17
SingleCellExperiment, 2, 4, 5, 7, 9, 11–15, 17

varianceStabilizingTransformation, 7, 8
vst, 7, 8