Package 'muscat'

January 24, 2025

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 1.20.0

Depends R (>= 4.4)

Imports BiocParallel, blme, ComplexHeatmap, data.table, DESeq2, dplyr, edgeR, ggplot2, glmmTMB, grDevices, grid, limma, lmerTest, lme4, Matrix, matrixStats, methods, progress, purrr, rlang, S4Vectors, scales, scater, scuttle, sctransform, stats, SingleCellExperiment, SummarizedExperiment, variancePartition, viridis

Suggests BiocStyle, countsimQC, ExperimentHub, iCOBRA, knitr, patchwork, phylogram, RColorBrewer, reshape2, rmarkdown, statmod, stageR, testthat, UpSetR

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

License GPL-3

VignetteBuilder knitr

RoxygenNote 7.3.2

Encoding UTF-8

URL https://github.com/HelenaLC/muscat

BugReports https://github.com/HelenaLC/muscat/issues

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

git_branch RELEASE_3_20

git_last_commit efa7f35

git_last_commit_date 2024-10-29

Repository Bioconductor 3.20

Date/Publication 2025-01-23

Author Helena L. Crowell [aut, cre] (<https://orcid.org/0000-0002-4801-1767>),
Pierre-Luc Germain [aut],
Charlotte Soneson [aut],
Anthony Sonrel [aut],
Jeroen Gilis [aut],
Davide Risso [aut],
Lieven Clement [aut],
Mark D. Robinson [aut, fnd]

Maintainer Helena L. Crowell <helena@crowell.eu>

Contents

`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", "prop.detected", "num.detected"),
  scale = FALSE,
  verbose = TRUE,
  BPPARAM = SerialParam(progressbar = verbose)
)
```

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. Passed to summarizeAssayByGroup.
scale	logical. Should pseudo-bulks be scaled with the effective library size & multiplied by 1M?
verbose	logical. Should information on progress be reported?
BPPARAM	a BiocParallelParam object specifying how aggregation should be parallelized.

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()$agg_pars, and the number of cells that were aggregated are accessible in int_colData()$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

```
# pseudobulk counts by cluster-sample
data(example_sce)
pb <- aggregateData(example_sce)

library(SingleCellExperiment)
assayNames(example_sce)  # one sheet per cluster
head(assay(example_sce)) # n_genes x n_samples

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

```
# aggregate by cluster only
pb <- aggregateData(example_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(example_sce)
library(SingleCellExperiment)

frq <- calcExprFreqs(example_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 `logNormCounts` 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

```
# 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,
    kid = "cell",
    sid = "id",
    gid = "stim",
    drop = TRUE)
```

---

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("dream2", "dream", "vst", "poisson", "nbinom", "hybrid"),
  n_cells = 10,
  n_samples = 2,
```

```
  min_count = 1,
  min_cells = 20,
  verbose = TRUE,
  BPPARAM = SerialParam(progressbar = verbose),
  vst = c("sctransform", "DESeq2"),
  ddf = c("Satterthwaite", "Kenward-Roger", "lme4"),
  dup_corr = FALSE,
  trended = FALSE,
  bayesian = FALSE,
  blind = TRUE,
  REML = TRUE,
  moderate = FALSE
)

.mm_dream(
  x,
  coef = NULL,
  covs = NULL,
  dup_corr = FALSE,
  trended = FALSE,
  ddf = c("Satterthwaite", "Kenward-Roger"),
  verbose = FALSE,
  BPPARAM = SerialParam(progressbar = verbose)
)

.mm_dream2(
  x,
  coef = NULL,
  covs = NULL,
  ddf = c("Satterthwaite", "Kenward-Roger"),
  verbose = FALSE,
  BPPARAM = SerialParam(progressbar = verbose)
)

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

.mm_glmm(
  x,
  coef = NULL,
  covs = NULL,
  family = c("poisson", "nbinom"),
```

```
  moderate = FALSE,
  verbose = TRUE,
  BPPARAM = SerialParam(progressbar = verbose)
)
```

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 "dream2" (default, lme4 with voom-weights), "dream" (previous implementation of the dream method), "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.
verbose	logical specifying whether messages on progress and a progress bar should be displayed.
BPPARAM	a BiocParallelParam object specifying how differential testing should be parallelized.
vst	method to use as variance-stabilizing transformations. "sctransform" for vst; "DESeq2" for varianceStabilizingTransformation.
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.
dup_corr	logical; whether to use duplicateCorrelation.
trended	logical; whether to use expression-dependent variance priors in eBayes.
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.
moderate	logical; whether to perform empirical Bayes moderation.
family	character string specifying which GLMM to fit: "poisson" for bglmer, "nbinom" for glmmTMB.

Details

The .mm_* functions (e.g. .mm_dream) 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 = "dream2"` `variancePartition`'s (>=1.14.1) voom-lme4-implementation of mixed models for RNA-seq data; function `dream`.

`method = "dream"` `variancePartition`'s older 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_dream2()`: see details.
- `.mm_vst()`: see details.
- `.mm_glmm()`: 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

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

# downsample to 100 genes
gs <- sample(nrow(sub), 100)
sub <- sub[gs, ]

# run DS analysis using cell-level mixed-model
res <- mmDS(sub, method = "dream", 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", "DD"),
  design = NULL,
  coef = NULL,
  contrast = NULL,
  min_cells = 10,
  filter = c("both", "genes", "samples", "none"),
  treat = FALSE,
  verbose = TRUE,
  BPPARAM = SerialParam(progressbar = verbose)
)

pbDD(
  pb,
  design = NULL,
  coef = NULL,
  contrast = NULL,
  min_cells = 10,
  filter = c("both", "genes", "samples", "none"),
  verbose = TRUE,
  BPPARAM = SerialParam(progressbar = verbose)
)
```

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. Can be a list for multiple, independent comparisons.
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.
filter	character string specifying whether to filter on genes, samples, both or neither.
treat	logical specifying whether empirical Bayes moderated-t p-values should be computed relative to a minimum fold-change threshold. Only applicable for methods "limma-x" (treat) and "edgeR" (glmTreat), and ignored otherwise.

verbose	logical. Should information on progress be reported?
BPPARAM	a BiocParallelParam object specifying how differential testing should be parallelized.

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(example_sce)

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

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

# count nb. of DE genes by cluster
vapply(res$table$stim, 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, function(u)
  filter(u, abs(logFC) > 1) %>%
    arrange(p_adj.loc) %>%
    slice(seq_len(5)))
```

pbFlatten *pbFlatten Flatten pseudobulk SCE*

Description

Flattens a pseudobulk SingleCellExperiment as returned by aggregateData such that all cell subpopulations are represented as a single assay.

Usage

```
pbFlatten(pb, normalize = TRUE)
```

Arguments

pb	a pseudobulk SingleCellExperiment as returned by aggregateData, with different subpopulations as assays.
normalize	logical specifying whether to compute a logcpm assay.

Value

a SingleCellExperiment.

Examples

```
data(example_sce)
library(SingleCellExperiment)
pb_stack <- aggregateData(example_sce)
(pb_flat <- pbFlatten(pb_stack))
ncol(pb_flat) == ncol(pb_stack)*length(assays(pb_stack))
```

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; `"none"` to retain original ordering.
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

```
# compute pseudobulks & run DS analysis
data(example_sce)
pb <- aggregateData(example_sce)
res <- pbDS(pb)

# cluster-sample expression means
```

```
pbHeatmap(example_sce, res)

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

# plot specific gene across all clusters
pbHeatmap(example_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(example_sce)
pb <- aggregateData(example_sce)
pbMDS(pb)
```

prepSCE *Prepare SCE for DS analysis*

Description

...

Usage

```
prepSCE(
  x,
  kid = "cluster_id",
  sid = "sample_id",
  gid = "group_id",
  drop = FALSE
)
```

Arguments

x	a SingleCellExperiment.
kid, sid, gid	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,g 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

# 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)
cd <- data.frame(group = gids, id = sids, cluster = kids, batch)

# construct SCE
library(scuttle)
sce <- mockSCE(ncells = nc, ngenes = ng)
colData(sce) <- cbind(colData(sce), cd)

# prep. for workflow
```

```
sce <- prepSCE(sce, kid = "cluster", sid = "id", gid = "group")
head(colData(sce))
metadata(sce)$experiment_info
sce
```

---

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

```
# estimate simulation parameters
data(example_sce)
ref <- prepSim(example_sce)

# tabulate number of genes/cells before vs. after
ns <- cbind(
  before = dim(example_sce),
  after = dim(ref))
rownames(ns) <- c("#genes", "#cells")
ns

library(SingleCellExperiment)
head(rowData(ref)) # gene parameters
head(colData(ref)) # 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("row", "col"),
  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

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

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

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

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

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

---

simData *simData*

Description

Simulation of complex scRNA-seq data

Usage

```
simData(
  x,
  ng = nrow(x),
  nc = ncol(x),
  ns = NULL,
  nk = NULL,
  probs = NULL,
  dd = TRUE,
  p_dd = diag(6)[1, ],
  paired = FALSE,
  p_ep = 0.5,
  p_dp = 0.3,
  p_dm = 0.5,
  p_type = 0,
  lfc = 2,
  rel_lfc = NULL,
  phylo_tree = NULL,
  phylo_pars = c(ifelse(is.null(phylo_tree), 0, 0.1), 3),
  force = FALSE
)
```

Arguments

x	a SingleCellExperiment.
ng	number of genes to simulate. Importantly, for the library sizes computed by prepSim (= exp(x$offset)) to make sense, the number of simulated genes should match with the number of genes in the reference. To simulate a reduced number of genes, e.g. for testing and development purposes, please set force = TRUE.
nc	number of cells to simulate.

ns	number of samples to simulate; defaults to as many as available in the reference to avoid duplicated reference samples. Specifically, the number of samples will be set to n = nlevels(x$sample_id) when dd = FALSE, n per group when dd, paired = TRUE, and floor(n/2) per group when dd = TRUE, paired = FALSE. When a larger number samples should be simulated, set force = TRUE.
nk	number of clusters to simulate; defaults to the number of available reference clusters (nlevels(x$cluster_id)).
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.
dd	whether or not to simulate differential distributions; if TRUE, two groups are simulated and ns corresponds to the number of samples per group, else one group with ns samples is simulated.
p_dd	numeric vector of length 6. Specifies the probability of a gene being EE, EP, DE, DP, DM, or DB, respectively.
paired	logical specifying whether a paired design should be simulated (both groups use the same set of reference samples) or not (reference samples are drawn at random).
p_ep, p_dp, p_dm	numeric specifying the proportion of cells to be shifted to a different expression state in one group (see details).
p_type	numeric. Probability 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 (logarithm base 2) 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.
phylo_tree	newick tree text representing cluster relations and their relative distance. An explanation of the syntax can be found here. The distance between the nodes, except for the original branch, will be translated in the number of shared genes between the clusters belonging to these nodes (this relation is controlled with phylo_pars). The distance between two clusters is defined as the sum of the branches lengths separating them.
phylo_pars	vector of length 2 providing the parameters that control the number of type genes. Passed to an exponential PDF (see details).
force	logical specifying whether to force simulation when ng and/or ns don't match the number of available reference genes and samples, respectively.

Details

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

The simulation of type genes can be performed in 2 ways; (1) via p_type to simulate independent clusters, OR (2) via phylo_tree to simulate a hierarchical cluster structure.

For (1), a subset of p_type % of genes are selected per cluster to use a different references genes than the remainder of clusters, giving rise to cluster-specific NB means for count sampling.

For (2), the number of shared/type genes at each node are given by a*G*e^(-b*d), where

- a – controls the percentage of shared genes between nodes. By default, at most 10% of the genes are reserved as type genes (when b = 0). However, it is advised to tune this parameter depending on the input prep_sce.
- b – determines how the number of shared genes decreases with increasing distance d between clusters (defined through phylo_tree).

Value

a SingleCellExperiment containing multiple clusters & samples across 2 groups as well as the following metadata:

cell metadata (colData(.)) a DataFrame containing, containing, for each cell, it's cluster, sample, and group ID.

gene metadata (rowData(.)) a DataFrame containing, for each gene, it's class (one of "state", "type", "none") and specificity (specs; NA for genes of type "state", otherwise a character vector of clusters that share the given gene).

experiment metadata (metadata(.)) experiment_info a data.frame summarizing the experimental design.

n_cells the number of cells for each sample.

gene_info a data.frame containing, for each gene in each cluster, it's differential distribution category, mean logFC (NA for genes for categories "ee" and "ep"), gene used as reference (sim_gene), dispersion sim_disp, and simulation means for each group sim_mean.A/B.

ref_sids/kidskids the sample/cluster IDs used as reference.

args a list of the function call's input arguments.

Author(s)

Helena L Crowell & Anthony Sonrel

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(example_sce)
library(SingleCellExperiment)

# prep. SCE for simulation
ref <- prepSim(example_sce)

# simulate data
(sim <- simData(ref, nc = 200,
  p_dd = c(0.9, 0, 0.1, 0, 0, 0),
  ng = 100, force = TRUE,
  probs = list(NULL, NULL, c(1, 0))))

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

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

# unbalanced sample sizes
sim <- simData(ref, nc = 100, ns = 2,
  probs = list(NULL, c(0.25, 0.75), NULL),
  ng = 10, force = TRUE)
table(sim$sample_id)

# one group only
sim <- simData(ref, nc = 100,
  probs = list(NULL, NULL, c(1, 0)),
  ng = 10, force = TRUE)
levels(sim$group_id)

# HIERARCHICAL CLUSTER STRUCTURE
# define phylogram specifying cluster relations
phylo_tree <- "(('cluster1':0.1,'cluster2':0.1):0.4,'cluster3':0.5);"
# verify syntax & visualize relations
library(phylogram)
plot(read.dendrogram(text = phylo_tree))

# let's use a more complex phylogeny
phylo_tree <- "(('cluster1':0.4,'cluster2':0.4):0.4,('cluster3':
  0.5,('cluster4':0.2,'cluster5':0.2,'cluster6':0.2):0.4):0.4);"
plot(read.dendrogram(text = phylo_tree))

# simulate clusters accordingly
sim <- simData(ref,
  phylo_tree = phylo_tree,
  phylo_pars = c(0.1, 3),
  ng = 500, force = TRUE)
# view information about shared 'type' genes
table(rowData(sim)$class)
```

`stagewise_DS_DD` *Perform two-stage testing on DS and DD analysis results*

Description

Perform two-stage testing on DS and DD analysis results

Usage

```
stagewise_DS_DD(res_DS, res_DD, sce = NULL, verbose = FALSE)
```

Arguments

`res_DS`	a list of DS testing results as returned by `pbDS` or `mmDS`.
`res_DD`	a list of DD testing results as returned by `pbDD` (or `pbDS` with `method="DD"`).

sce	(optional) SingleCellExperiment object containing the data that underlies testing, prior to summarization with aggregateData. Used for validation of inputs in order to prevent unexpected failure/results.
verbose	logical. Should information on progress be reported?

Value

A list of DFrames containing results for each contrast and cluster. Each table contains DS and DD results for genes shared between analyses, as well as results from stagewise testing analysis, namely:

- p_adj: FDR adjusted p-values for the screening hypothesis that a gene is neither DS nor DD (see ?stageR::getAdjustedPValues for details)
- p_val.DS/D: confirmation stage p-values for DS/D

Examples

```
data(example_sce)

pbs_sum <- aggregateData(example_sce, assay="counts", fun="sum")
pbs_det <- aggregateData(example_sce, assay="counts", fun="num.detected")

res_DS <- pbDS(pbs_sum, min_cells=0, filter="none", verbose=FALSE)
res_DD <- pbDD(pbs_det, min_cells=0, filter="none", verbose=FALSE)

res <- stagewise_DS_DD(res_DS, res_DD)
head(res[[1]][[1]]) # results for 1st cluster
```

Index