Package 'simPIC'

June 5, 2026

Version 1.9.0

Date 2023-02-02

Type Package

Title Flexible simulation of paired-insertion counts for single-cell ATAC-sequencing data

Author Sagrika Chugh [aut, cre] (<https://orcid.org/0000-0002-8050-5214>), Heejung Shim [aut], Davis McCarthy [aut]

Maintainer Sagrika Chugh <sagrika.chugh@gmail.com>

Depends R (>= 4.5.0), SingleCellExperiment

Imports BiocGenerics, checkmate (>= 2.0.0), fitdistrplus, matrixStats, actuar, Matrix, stats, SummarizedExperiment, rlang, S4Vectors, GenomeInfoDb, methods, scales, scuttle, edgeR, withr

Description simPIC is a package for simulating single-cell ATAC-seq count data. It provides a user-friendly, well documented interface for data simulation. Functions are provided for parameter estimation, realistic scATAC-seq data simulation, and comparing real and simulated datasets.

biocViews SingleCell, ATACSeq, Software, Sequencing, ImmunoOncology, DataImport

License GPL-3

Encoding UTF-8

Suggests bluster, ggplot2 (>= 3.4.0), knitr, rmarkdown, BiocStyle, testthat (>= 3.0.0), scater, scran, magick, splatter, VariantAnnotation, IRanges, GenomicRanges, preprocessCore

VignetteBuilder knitr

RoxygenNote 7.3.1

Config/testthat/edition 3

URL https://github.com/sagrikachugh/simPIC

BugReports https://github.com/sagrikachugh/simPIC/issues

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

git_branch devel

git_last_commit 9c476fc

git_last_commit_date 2026-05-28

Repository Bioconductor 3.24

Date/Publication 2026-06-04

Contents

`simPIC-package` *simPIC: Flexible simulation of paired-insertion counts for single-cell ATAC-sequencing data*

Description

simPIC is a package for simulating single-cell ATAC-seq count data. It provides a user-friendly, well documented interface for data simulation. Functions are provided for parameter estimation, realistic scATAC-seq data simulation, and comparing real and simulated datasets.

- count class (`newsimPICcount`)
- estimate (`simPICestimate`)
- simulate (`simPICsimulate`)
- plots (`simPICcompare`)

Author(s)

Maintainer: Sagrika Chugh <sagrika.chugh@gmail.com> (ORCID)

Authors:

- Heejung Shim
- Davis McCarthy

See Also

Useful links:

- https://github.com/sagrikachugh/simPIC
- Report bugs at https://github.com/sagrikachugh/simPIC/issues

`addFeatureStats` *Add feature statistics*

Description

Add additional feature statistics to a SingleCellExperiment object

Usage

```
addFeatureStats(
  sce,
  value = "counts",
  log = FALSE,
  offset = 1,
  no.zeros = FALSE
)
```

Arguments

sce	SingleCellExperiment to add feature statistics to.
value	the count value to calculate statistics.
log	logical. Whether to take log2 before calculating statistics.
offset	offset to add to avoid taking log of zero.
no.zeros	logical. Whether to remove all zeros from each feature before calculating statistics.

Details

Currently adds the following statistics: mean and variance. Statistics are added to the `rowData` slot and are named `Stat[Log]Value[No0]` where `Log` and `No0` are added if those arguments are true.

Value

SingleCellExperiment with additional feature statistics

---

convert_to_SCE *Convert Sparse Matrix to SingleCellExperiment object*

Description

This function converts a sparse matrix into a SingleCellExperiment (SCE) object.

Usage

```
convert_to_SCE(sparse_data)
```

Arguments

sparse_data	A sparse matrix containing count data, where rows are peaks and columns represent cells.

Value

A SingleCellExperiment (SCE) object with the sparse matrix stored in the "counts" assay.

`ensureCountsFirst` *Ensure counts assay is first*

Description

Reorders assays in a SingleCellExperiment so that the `counts` assay appears first. This keeps `assay(sce)` behavior predictable for downstream code that still assumes the primary assay is counts.

Usage

```
ensureCountsFirst(sce)
```

Arguments

`sce`	SingleCellExperiment object.

Value

SingleCellExperiment with counts assay first if present.

`getCounts` *Get counts from Single Cell Experiment object*

Description

Get counts matrix from a SingleCellExperiment object. If counts is missing a warning is issued and the first assay is returned.

Usage

```
getCounts(sce)
```

Arguments

`sce`	SingleCellExperiment object

Value

counts matrix

getLNormFactors *Get accessibility factors*

Description

Randomly generate multiplication factors from a log-normal distribution.

Usage

```
getLNormFactors(n.facs, sel.prob, neg.prob, fac.loc, fac.scale)
```

Arguments

n.facs	Number of factors to generate.
sel.prob	Probability that a factor will be selected to be different from 1.
neg.prob	Probability that a selected factor is less than one.
fac.loc	Location parameter for the log-normal distribution.
fac.scale	Scale factor for the log-normal distribution.

Value

Vector containing generated factors.

global *simPIC: Simulate single-cell ATAC-seq data*

Description

simPIC: Simulate single-cell ATAC-seq data

Value

globalvariables

newsimPICcount *newsimPICcount*

Description

Create a newsimPICcount object to store parameters.

Usage

```
newsimPICcount(...)
```

Arguments

...	Variables to set newsimPICcount object parameters.

Details

This function creates the object variable which is passed in all functions.

Value

new object from class simPICcount.

Examples

```
object <- newsimPICcount()
```

plot_theme *Custom theme for ggplot2*

Description

This function defines a custom theme for ggplot2 to ensure consistent visual appearance across multiple plots.

Usage

```
plot_theme()
```

Value

A ggplot2 theme object with predefined settings.

rbindMatched *Bind rows (matched)*

Description

Bind the rows of two data frames, keeping only the columns that are common to both.

Usage

```
rbindMatched(df1, df2)
```

Arguments

df1	first data.frame to bind.
df2	second data.frame to bind.

Value

data.frame containing rows from df1 and df2 but only common columns.

selectFit *Select fit*

Description

Trying two fitting methods and selecting the best one.

Usage

```
selectFit(data, distr, verbose = TRUE)
```

Arguments

data	The data to fit.
distr	Name of the distribution to fit.
verbose	Logical. Whether to print progress messages.

Details

The distribution is fitted to the data using each of the fitdist fitting methods. The fit with the smallest Cramer-von Mises statistic is selected.

Value

The selected fit object

setsimPICparameters *Set simPIC parameters*

Description

Set input parameters of the simPICcount object.

Usage

```
setsimPICparameters(object, update = NULL, ...)
```

Arguments

object	input simPICcount object.
update	new parameters.
...	set new parameters for simPICcount object.

Value

simPICcount object with updated parameters.

Examples

```
object <- newsimPICcount()
object <- setsimPICparameters(object, nCells = 200, nPeaks = 500)
```

simPICcompare *Compare SingleCellExperiment objects*

Description

Combine data from several SingleCellExperiment objects and produce some basic plots comparing them.

Usage

```
simPICcompare(
  sces,
  point.size = 0.2,
  point.alpha = 0.1,
  fits = TRUE,
  colours = NULL
)
```

Arguments

sces	named list of SingleCellExperiment objects to combine and compare.
point.size	size of points in scatter plots.
point.alpha	opacity of points in scatter plots.
fits	whether to include fits in scatter plots.
colours	vector of colours to use for each dataset.

Details

The returned list has three items: The first dataset in sces is treated as the reference dataset when computing Kolmogorov-Smirnov (KS) summaries.

RowData Combined row data from the provided SingleCellExperiments.

ColData Combined column data from the provided SingleCellExperiments.

KS Data frame summarizing KS statistics for peak means, library sizes, and cell sparsity, comparing each dataset to the first dataset in sces.

Plots Comparison plots

- Means Boxplot of mean distribution.
- Variances Boxplot of variance distribution.
- MeanVar Scatter plot with fitted lines showing the mean-variance relationship.
- LibrarySizes Boxplot of the library size distribution.
- ZerosPeak Boxplot of the percentage of each peak that is zero.
- ZerosCell Boxplot of the percentage of each cell that is zero.
- MeanZeros Scatter plot with fitted lines showing the mean-zeros relationship.

The plots returned by this function are created using ggplot and are only a sample of the kind of plots you might like to consider. The data used to create these plots is also returned and should be in the correct format to allow you to create further plots using ggplot.

Value

List containing the combined datasets and plots.

Examples

```
sim1 <- simPICsimulate(
    nPeaks = 1000, nCells = 500,
    pm.distr = "weibull", seed = 7856
)
sim2 <- simPICsimulate(
    nPeaks = 1000, nCells = 500,
    pm.distr = "gamma", seed = 4234
)
comparison <- simPICcompare(list(weibull = sim1, gamma = sim2))
names(comparison)
names(comparison$Plots)
```

simPICcount *The simPICcount class*

Description

S4 class that holds parameters for simPIC simulation.

Value

a simPIC class object. The parameters not shown in brackets can be estimated from real data using `simPICestimate`. For details of the simPIC simulation see `simPICsimulate`. The default parameters are based on the PBMC10k dataset and can be reproduced using the test data and script provided in `inst/scripts`.

Parameters

simPIC simulation parameters:

`nPeaks` The number of peaks to simulate.

`nCells` The number of cells to simulate.

`[seed]` Seed to use for generating random numbers.

`[default]` Logical value indicating whether to use default parameters (TRUE) or learn parameters from data (FALSE).

Library size parameters `lib.size.meanlog` meanlog (location) parameter for the library size log-normal distribution.

`lib.size.sdlog` sdlog (scale) parameter for the library size log-normal distribution.

Peak mean parameters `mean.scale` scale parameter for the mean weibull distribution.

`mean.shape` shape parameter for the mean weibull distribution.

Cell sparsity parameters `sparsity` Probability that contributes to the sparsity of the final simulated matrix.

simPICEstBCV *Estimate simPIC Biological Coefficient of Variation parameters*

Description

Parameters are estimated using the `estimateDisp` function in the `edgeR` package.

Usage

```
simPICEstBCV(counts, object, verbose)
```

Arguments

`counts`	counts matrix to estimate parameters from.
`object`	simPICcount object to store estimated values in.
`verbose`	Logical. Whether to print progress messages.

Details

The `estimateDisp` function is used to estimate the common dispersion and prior degrees of freedom. See `estimateDisp` for details. When estimating parameters on simulated data we found a broadly linear relationship between the true underlying common dispersion and the edgeR estimate, therefore we apply a small correction, `disp = -0.3 + 0.15 * edgeR.disp`.

Value

simPICcount object with estimated values.

`simPICestimate` *Estimate simPIC simulation parameters*

Description

Estimate simulation parameters for library size, peak means, and sparsity from a real peak-by-cell input matrix.

Usage

```
simPICestimate(
  counts,
  object = newsimPICcount(),
  pm.distr = c("lngamma", "gamma", "weibull", "pareto"),
  method = c("single", "groups"),
  verbose = TRUE
)

## S3 method for class 'SingleCellExperiment'
simPICestimate(
  counts,
  object = newsimPICcount(),
  pm.distr = "lngamma",
  method = "single",
  verbose = TRUE
)

## S3 method for class 'dgCMatrix'
simPICestimate(
  counts,
  object = newsimPICcount(),
  pm.distr = "lngamma",
  method = "single",
  verbose = TRUE
)
```

Arguments

`counts`	either a sparse peak by cell count matrix, or a SingleCellExperiment object containing count data to estimate parameters.
`object`	simPICcount object to store estimated parameters and counts.

pm.distr	statistical distribution for estimating peak mean parameters. Available distributions: gamma, weibull, lngamma, pareto. Default is lngamma.
method	Simulation mode. Use "single" to estimate parameters for one cell type or "groups" for distinct cell types.
verbose	logical variable. Prints the simulation progress if TRUE.

Value

simPICcount object containing all estimated parameters.

Examples

```
counts <- readRDS(system.file("extdata", "test.rds", package = "simPIC"))
est <- newsimPICcount()
est <- simPICestimate(counts, pm.distr = "lngamma")
```

---

simPICestimateLibSize *Estimate simPIC library size parameters*

Description

Estimate the library size parameters for simPIC simulation.

Usage

```
simPICestimateLibSize(counts, object, verbose)
```

Arguments

counts	count matrix.
object	simPICcount object to store estimated values.
verbose	Logical. Whether to print progress messages.

Details

Parameters for the lognormal distribution are estimated by fitting the library sizes using fitdist. All the fitting methods are tried and the fit with the best Cramer-von Mises statistic is selected.

Value

simPICcount object with estimated library size parameters.

simPICestimatePeakMean

Estimate simPIC peak means

Description

Estimate peak mean parameters for simPIC simulation

Usage

```
simPICestimatePeakMean(norm.counts, object, pm.distr, verbose)
```

Arguments

norm.counts	Library-size normalized count matrix.
object	simPICcount object to store estimated values.
pm.distr	distribution parameter for peak means.
verbose	Logical. Whether to print progress messages.

Details

Parameters for gamma distribution are estimated by fitting the mean normalized counts using `fitdist`. All the fitting methods are tried and the fit with the best Cramer-von Mises statistic is selected.

Value

simPICcount object containing all estimated parameters

simPICestimateSparsity

Estimate sparsity

Description

This function estimates cell sparsity from a normalized count matrix and updates the parameters of a simPIC object accordingly.

Usage

```
simPICestimateSparsity(norm.counts, object, verbose)
```

Arguments

norm.counts	A normalized count matrix to estimate parameters from.
object	simPICcount object to store estimated parameters.
verbose	Logical. Whether to print progress messages.

Value

simPICcount object with updated sparsity parameter.

simPICget *Get a single simPICcount parameter*

Description

Get the value of a single variable from input simPICcount object.

Usage

```
simPICget(object, name)
```

Arguments

object	input simPICcount object.
name	name of the parameter.

Value

Value of the input parameter.

Examples

```
object <- newsimPICcount()
nPeaks <- simPICget(object, "nPeaks")
```

simPICgetparameters *Get parameters*

Description

Get multiple parameter values from a simPIC object.

Usage

```
simPICgetparameters(object, names)
```

Arguments

object	input object to get values from.
names	vector of names of the parameters to get.

Value

List with the values of the selected parameters.

Examples

```
object <- newsimPICcount()
simPICgetparameters(object, c("nPeaks", "nCells", "peak.mean.shape"))
```

simPICMicrogliaExample

Run the Packaged Microglia splatPop Example

Description

Run a streamlined population-scale Microglia example using packaged `SingleCellExperiment`, sample mapping, peak annotation, and VCF files. The workflow follows the splatPop estimation pattern by estimating population means from retained donor-batch units while estimating single-cell behaviour from the donor-batch unit with the most cells.

Usage

```
simPICMicrogliaExample(
  sce = NULL,
  sample.map = NULL,
  peak.annot = NULL,
  vcf = NULL,
  params = NULL,
  min.cells = 20L,
  pop.cv.bins = 10L,
  eqtl.n = 0.05,
  similarity.scale = 2,
  eqtl.dist = 1e+08,
  sparsify = FALSE,
  pca.ntop = 100L,
  pca.components = 5L,
  plot.n.samples = 4L,
  plot.samples = NULL,
  comparison.batch = NULL,
  comparison.pathology = NULL,
  seed = NULL,
  verbose = TRUE,
  ...
)
```

Arguments

`sce`	Optional `SingleCellExperiment` or path to an RDS file containing the packaged Microglia SCE. If `NULL`, the packaged example is used.
`sample.map`	Optional data.frame or path to a tab-separated sample map linking VCF donor IDs to Microglia donor IDs. If `NULL`, the packaged example is used.
`peak.annot`	Optional data.frame or path to the chr14 peak annotation table. If `NULL`, the packaged example is used.
`vcf`	Optional `CollapsedVCF` or path to the chr14 VCF used for the Microglia example. If `NULL`, the packaged example is used.
`params`	Optional `SplatPopParams`. If `NULL`, a new object is created with `pop.cv.bins`.
`min.cells`	Minimum number of cells required per donor-batch unit for inclusion in the aggregated population means. Units must exceed this threshold.

pop.cv.bins	Number of CV bins used when params is created internally.
eqtl.n	Proportion of peaks to simulate with caQTL effects. The packaged example uses a modest default to keep the workflow lightweight while still including genetic effects.
similarity.scale	Similarity scale passed to splatter::setParams() before simulation.
eqtl.dist	Maximum cis-distance used when assigning caQTL effects.
sparsify	Logical. Whether to sparsify the simulated output.
pca.ntop	Number of variable peaks used for PCA plots.
pca.components	Number of principal components to compute for the PCA summaries.
plot.n.samples	Number of libraries to show in the default PCA comparison panels. The top libraries are chosen by real-data cell count.
plot.samples	Optional character vector of specific libraries to show in the PCA comparison panels. When NULL, the top plot.n.samples libraries are used.
comparison.batch	Optional real-data batch/library label used for the Poptrial-style comparison panels. When supplied, the real data are subset to this batch and the simulated data are matched by the same sample IDs.
comparison.pathology	Optional pathology label used together with comparison.batch for Poptrial-style panels, for example "earlyAD".
seed	Optional random seed.
verbose	Logical. Whether to print progress messages.
...	Additional parameters passed to splatter::setParams() prior to simulation.

Value

A list containing:

sce Filtered real-data Microglia SingleCellExperiment.

bigcounts The donor-batch subset with the most cells used for count-based estimation.

aggregated Aggregated donor-batch means as a SingleCellExperiment.

params Estimated SplatPopParams after Microglia-specific parameter updates.

sim Simulated SingleCellExperiment.

plot_samples Libraries used for the default PCA comparison panels.

plots A named list containing real and simulated PCA plot summaries, side-by-side comparison panels, and bluster-based silhouette and neighborhood-purity panels.

sample_map The cleaned sample map used for alignment.

peak_annot The chr14 peak annotation used for simulation.

vcf The aligned VCF used for simulation.

Examples

```
if (requireNamespace("splatter", quietly = TRUE) &&
    requireNamespace("VariantAnnotation", quietly = TRUE) &&
    requireNamespace("bluster", quietly = TRUE)) {
out <- simPICMicrogliaExample(verbose = FALSE)
out$plot_samples
names(out$plots$comparison)
}
```

simPICplotBlusterComparison *Plot Poptrial-Style PCA, Silhouette, and Purity Comparisons*

Description

Create side-by-side real-versus-simulated comparison panels inspired by the bluster-based plotting workflow used in `Poptrial.Rmd`. The comparison can be restricted either to a chosen set of samples or to a real-data batch, in which case the simulated data are matched by the same sample IDs.

Usage

```
simPICplotBlusterComparison(
  real.sce,
  simulated.sce,
  sample.col = "Sample",
  batch.col = NULL,
  subset.batch = NULL,
  pathology.col = NULL,
  subset.pathology = NULL,
  plot.samples = NULL,
  plot.n.samples = 5L,
  pca.ntop = 2000L,
  pca.components = 10L,
  point.size = 0.8,
  verbose = TRUE
)
```

Arguments

`real.sce`	Real-data `SingleCellExperiment`.
`simulated.sce`	Simulated `SingleCellExperiment`.
`sample.col`	Column in `colData()` containing sample IDs.
`batch.col`	Optional column in the real-data `colData()` used to define a batch or library subset such as `"Library5"`.
`subset.batch`	Optional batch/library value used to subset the real data before matching the simulated samples.
`pathology.col`	Optional pathology column in the real-data `colData()`.
`subset.pathology`	Optional pathology value used together with `subset.batch` to restrict the real-data comparison subset.
`plot.samples`	Optional character vector of samples to compare. Ignored when `subset.batch` is supplied.
`plot.n.samples`	Number of top samples to keep when neither `subset.batch` nor `plot.samples` is supplied.
`pca.ntop`	Number of most variable peaks used for PCA.
`pca.components`	Number of principal components to compute.
`point.size`	Point size used in PCA panels.
`verbose`	Logical. Whether to print progress messages.

Value

A list with the subsetted real and simulated SCEs, the selected sample IDs, the bluster metric tables, and three ggplot panels: `cell_pca_by_sample`, `silhouette_width`, and `neighborhood_purity`.

`simPICplotPopulationPCA` *Plot Population-Style PCA Summaries for Real Data*

Description

Create splatPop-style PCA plots from a real peak-by-cell matrix or `SingleCellExperiment`. When metadata is not already available, sample IDs can be inferred from cell names, which is useful for peak-by-cell matrices where columns are encoded like `sample#barcode`.

Usage

```
simPICplotPopulationPCA(
  counts,
  sample.col = NULL,
  batch.col = NULL,
  sample.pattern = "#.*$",
  sample.replacement = "",
  pca.ntop = 2000,
  pca.components = 10,
  aggregate.by.sample = TRUE,
  point.size = 0.8,
  verbose = TRUE
)
```

Arguments

`counts`	A peak-by-cell count matrix, sparse `Matrix`, or `SingleCellExperiment`.
`sample.col`	Optional name of a sample column already present in `colData(counts)`. If `NULL`, sample IDs are inferred from column names using `sample.pattern`.
`batch.col`	Optional name of a batch column already present in `colData(counts)`.
`sample.pattern`	Regular expression used to strip the barcode suffix from cell names when inferring sample IDs.
`sample.replacement`	Replacement string used with `sample.pattern`.
`pca.ntop`	Number of most variable peaks to use for PCA.
`pca.components`	Number of principal components to compute.
`aggregate.by.sample`	Logical. Whether to also aggregate cells by sample and create a donor-level PCA plot.
`point.size`	Point size passed to `scater::plotPCA()`.
`verbose`	Logical. Whether to print progress messages.

Value

A list containing:

`cell_sce` Cell-level `SingleCellExperiment` with inferred metadata, log-normalized counts, and PCA.

`sample_sce` Sample-aggregated `SingleCellExperiment` if `aggregate.by.sample = TRUE`, otherwise `NULL`.

`plots` A named list of `ggplot` objects.

Examples

```
if (requireNamespace("scater", quietly = TRUE)) {
    counts <- matrix(rpois(50 * 24, lambda = 3), nrow = 50, ncol = 24)
    colnames(counts) <- paste0(
        rep(paste0("Sample", 1:6), each = 4),
        "#Cell",
        seq_len(ncol(counts))
    )
    plots <- simPICplotPopulationPCA(counts, verbose = FALSE)
    names(plots$plots)
}
```

---

simPICsimBatchCellMeans

Simulate batch means

Description

Simulate a mean for each peak in each cell incorporating batch effect factors.

Usage

```
simPICsimBatchCellMeans(object, sim)
```

Arguments

`object`	simPICcount object with simulation parameters.
`sim`	SingleCellExperiment to add batch means to.

Value

SingleCellExperiment with simulated batch means.

simPICsimBatchEffects *Simulate batch effects*

Description

Simulate batch effects. Batch effect factors for each batch are produced using `getLNormFactors` and these are added along with updated means for each batch.

Usage

```
simPICsimBatchEffects(object, sim)
```

Arguments

`object`	simPICcount object with simulation parameters.
`sim`	SingleCellExperiment to add batch effects to.

Value

SingleCellExperiment with simulated batch effects.

simPICsimCellMeans *Simulate cell means*

Description

Simulate a peak by cell matrix given the mean accessibility for each peak in each cell. Cells start with the mean accessibility for the group they belong to (when simulating groups). The selected means are adjusted for each cell's expected library size.

Usage

```
simPICsimSingleCellMeans(object, sim)

simPICsimulateGroupCellMeans(object, sim)
```

Arguments

`object`	simPIC object with simulation parameters.
`sim`	SingleCellExperiment to add cell means to.

Value

SingleCellExperiment with added cell means.

`simPICsimulate` *simPIC simulation*

Description

Simulate a peak-by-cell count matrix using simPIC methods.

Usage

```
simPICsimulate(
  object = newsimPICcount(),
  pm.distr = "lngamma",
  method = c("single", "groups"),
  verbose = TRUE,
  ...
)

simPICsimulatesingle(object = newsimPICcount(), verbose = TRUE, ...)

simPICsimulatemulti(
  object = newsimPICcount(),
  pm.distr = "lngamma",
  method = c("groups"),
  verbose = TRUE,
  ...
)
```

Arguments

`object`	simPICcount object with simulation parameters. See `simPICcount` for details.
`pm.distr`	distribution parameter for peak means. Available distributions: gamma, weibull, lngamma, pareto. Default is lngamma.
`method`	Simulation mode. Use `"single"` to simulate one cell type or `"groups"` to simulate distinct cell types.
`verbose`	Logical. Whether to print progress messages.
`...`	Any additional parameter settings to override what is provided in `simPICcount` object.

Details

simPIC provides the option to manually adjust each of the `simPICcount` object parameters by calling `setsimPICparameters`.

The simulation involves the following steps:

1. Set up simulation parameters
2. Set up SingleCellExperiment object
3. Simulate library sizes
4. Simulate sparsity
5. Simulate peak means

6. Create final synthetic counts

The final output is a SingleCellExperiment object that contains the simulated count matrix. The parameters are stored in the colData (for cell-specific information), rowData (for peak-specific information), or assays (for peak-by-cell matrices).

Value

SingleCellExperiment object containing the simulated counts.

Examples

```
# default simulation
sim <- simPICsimulate(pm.distr = "lngamma")
```

`simPICsimulateBCVMeans` *Simulate BCV means*

Description

Simulate means for each peak in each cell that are adjusted to follow a mean-variance trend using Biological Coefficient of Variation taken from and inverse gamma distribution.

Usage

```
simPICsimulateBCVMeans(object, sim)
```

Arguments

`object`	simPICcount object with simulation parameters.
`sim`	SingleCellExperiment to add BCV means to.

Value

SingleCellExperiment with simulated BCV means.

`simPICsimulateLibSize` *Simulate simPIC library sizes*

Description

Generate library sizes for cells in simPIC simulation based on the estimated values of mus and sigmas.

Usage

```
simPICsimulateLibSize(object, sim, verbose)
```

Arguments

object	simPICcount object with simulation parameters.
sim	SingleCellExperiment object containing simulation parameters.
verbose	Logical. Whether to print progress messages.

Value

SingleCellExperiment object with simulated library sizes.

---

simPICsimulatemultiDA *Simulate group differential accessibility*

Description

Simulate differential accessibility. Differential accessibility factors for each group are produced using getLNormFactors and these are added along with updated means for each group. For paths care is taken to make sure they are simulated in the correct order.

Usage

```
simPICsimulatemultiDA(object, sim)
```

Arguments

object	simPICcount object with simulation parameters.
sim	SingleCellExperiment to add differential accessibility to.

Value

SingleCellExperiment with simulated differential accessibility.

---

simPICsimulatePeakMean *Simulate simPIC peak means*

Description

Generate peak means for cells in simPIC simulation based on the estimated values of shape and rate parameters.

Usage

```
simPICsimulatePeakMean(object, sim, pm.distr, verbose)
```

Arguments

object	simPICcount object with simulation parameters.
sim	SingleCellExperiment object containing simulation parameters.
pm.distr	distribution parameter for peak means. Available distributions: gamma, weibull, lngamma, pareto. Default is lngamma.
verbose	logical. Whether to print progress messages.

Value

SingleCellExperiment object with simulated peak means.

---

simPICsimulateTrueCounts

Simulate true counts

Description

Counts are simulated from a Poisson distribution where each peak has a mean, expected library size and proportion of accessible chromatin.

Usage

```
simPICsimulateTrueCounts(object, sim)
```

Arguments

object	simPICcount object with simulation parameters.
sim	SingleCellExperiment object containing simulation parameters.

Value

SingleCellExperiment object with simulated true counts.

---

simPICsimulateTrueCountsGroups

Simulate true counts groups

Description

Counts are simulated from a Poisson distribution where each peak has a mean, expected library size and proportion of accessible chromatin.

Usage

```
simPICsimulateTrueCountsGroups(object, sim)
```

Arguments

object	simPICcount object with simulation parameters.
sim	SingleCellExperiment object containing simulation parameters.

Value

SingleCellExperiment object with simulated true counts.

---

splatPopEstimatePeak *Estimate Peak-Based splatPop Parameters*

Description

Estimate splatPop population parameters from peak-by-cell counts. This is a peak-based wrapper around `splatter::splatPopEstimate()` that can optionally aggregate donor-level peak means from a `SingleCellExperiment` object and then apply the simPIC BCV correction.

Usage

```
splatPopEstimatePeak(
  counts = NULL,
  means = NULL,
  eqtl = NULL,
  params = NULL,
  sample.col = "Sample",
  batch.col = NULL,
  aggregate.by = c("sample", "sample_batch"),
  min.cells = 1,
  pop.cv.bins = 50,
  apply.bcv.correction = TRUE,
  verbose = TRUE
)
```

Arguments

counts	Either a `SingleCellExperiment` object or a peak-by-cell count matrix. When `means` is `NULL`, `counts` must be a `SingleCellExperiment` with sample-level metadata so donor means can be aggregated automatically.
means	Optional dense matrix of aggregated peak means, with peaks in rows and donors/samples in columns.
eqtl	Optional empirical eQTL table to pass through to `splatter::splatPopEstimate()`.
params	Optional `SplatPopParams` object. If `NULL`, a new object is created with `pop.cv.bins`.
sample.col	Name of the sample/donor column in `colData(counts)` when `counts` is a `SingleCellExperiment`.
batch.col	Optional batch column in `colData(counts)` used when `aggregate.by = "sample_batch"`.
aggregate.by	Whether automatically derived means should be aggregated by sample or by sample-batch combinations.
min.cells	Minimum number of cells required per aggregation unit when deriving `means` from a `SingleCellExperiment`.

pop.cv.bins Number of CV bins to use when params is created internally.

apply.bcv.correction
Logical. If TRUE, apply the simPIC BCV correction rule directly to the estimated SplatPopParams.

verbose Logical. Whether to print progress messages.

Value

A SplatPopParams object.

Examples

```
if (requireNamespace("splatter", quietly = TRUE) &&
    requireNamespace("VariantAnnotation", quietly = TRUE)) {
    set.seed(101)
    gene_means <- rgamma(60, shape = 2, rate = 0.4)
    cell_scales <- runif(48, min = 0.7, max = 1.4)
    counts <- vapply(
        cell_scales,
        function(scale) {
            rnbinom(60, mu = gene_means * scale, size = 0.2)
        },
        numeric(60)
    )
    sce <- SingleCellExperiment::SingleCellExperiment(
        assays = list(counts = counts),
        colData = data.frame(
            Sample = rep(paste0("S", 1:12), each = 4),
            Batch = rep(c("B1", "B2"), each = 24)
        )
    )
    params <- splatPopEstimatePeak(
        counts = sce,
        sample.col = "Sample",
        batch.col = "Batch",
        aggregate.by = "sample",
        min.cells = 2,
        pop.cv.bins = 10,
        verbose = FALSE
    )
    params
}
```

---

splatPopSimulatePeak *Simulate Peak-Based Population Data with splatPop*

Description

Simulate peak-based single-cell data using splatter::splatPopSimulate while coercing peak annotations into the GFF-like structure that splatPop expects internally.

Usage

```
splatPopSimulatePeak(
  params = NULL,
  vcf = NULL,
  method = c("single", "groups", "paths"),
  gff = NULL,
  eqtl = NULL,
  means = NULL,
  key = NULL,
  counts.only = FALSE,
  sparsify = TRUE,
  verbose = TRUE,
  ...
)
```

Arguments

params	Optional SplatPopParams object. If NULL, a new object is created.
vcf	Optional VariantAnnotation VCF object. If NULL, the splatter mock VCF is used.
method	Simulation mode passed through to splatPopSimulate().
gff	Peak annotation supplied either as NULL, a GFF-like data.frame, or a GRanges. Peak annotations are coerced to a GFF-like table and stored as gene-like features for the underlying splatPop machinery.
eqtl	Optional empirical eQTL table.
means	Optional empirical population means matrix.
key	Optional splatPop key.
counts.only	Logical. Whether to keep only counts in the simulated object.
sparsify	Logical. Whether to sparsify the resulting assays.
verbose	Logical. Whether to print progress messages.
...	Additional parameters passed to splatter::splatPopSimulate.

Value

A SingleCellExperiment object.

Examples

```
if (requireNamespace("splatter", quietly = TRUE) &&
    requireNamespace("VariantAnnotation", quietly = TRUE)) {
    params <- splatter::newSplatPopParams(nGenes = 20)
    params <- splatter::setParams(params, batchCells = c(20))
    sim <- splatPopSimulatePeak(
        params = params,
        vcf = splatter::mockVCF(),
        gff = splatter::mockGFF()[seq_len(20), ],
        sparsify = FALSE,
        verbose = FALSE
    )
    sim
}
```

Index