

Package: scMaSigPro (via r-universe)

October 28, 2026

Type Package

Title Differential Expression Analysis Along Single-Cell Trajectories

Version 0.6.0

Description Identifies genes whose expression changes along single-cell pseudotime and differs between the branching paths of a trajectory. Cells are pseudo-bulked into pseudotime bins that preserve their ordering, and the polynomial generalised linear model of the 'maSigPro' package is fitted to the binned counts with a negative binomial family and size-factor offsets. Methods are described in Srivastava et al. (2024) [doi:10.1093/bioinformatics/btae443](https://doi.org/10.1093/bioinformatics/btae443).

License GPL (>= 2)

Imports methods, stats, Matrix, S4Vectors (>= 0.40.2), SummarizedExperiment, SingleCellExperiment (>= 1.24.0), MASS, ggplot2, BiocParallel

Depends R (>= 4.6.0)

Encoding UTF-8

Language en-GB

Roxygen list(markdown = TRUE)

Suggests testthat, roxygen2, knitr, rmarkdown, BiocStyle, UpSetR, patchwork, mclust, rlang, pkgload, maSigPro (>= 1.74.0), igraph, slingshot, DelayedMatrixStats

Config/testthat/edition 3

URL <https://github.com/BioBam/scMaSigPro/>,
<https://spriyansh.github.io/scMaSigPro/>

biocViews Clustering, Regression, TimeCourse, DifferentialExpression, GeneExpression, RNASeq, Sequencing, Software, SingleCell, Transcriptomics, MultipleComparison, Visualization

BugReports <https://github.com/BioBam/scMaSigPro/issues>

VignetteBuilder knitr

Config/roxygen2/version 8.1.0
Config/pak/sysreqs zlib1g-dev
Repository <https://spriyansh.r-universe.dev>
Date/Publication 2026-10-28 06:55:07 UTC
RemoteUrl <https://github.com/spriyansh/scMaSigPro>
RemoteRef HEAD
RemoteSha f7b543ea6091ac73db96185a10dc625866b1c08d

Contents

as_scmp	3
assign_paths	4
cDense	6
create_scmp	8
cSparse	9
eDense	10
eSparse	11
Estimates	13
MatrixDesign	14
multi.lin.sce	15
ParameterConfig	16
pathAssign	17
paths_from_graph	18
pb_counts	20
plotBinTile	21
plotDiagnostics	22
plotIntersect	23
plotTrend	24
plotTrendCluster	26
predictors	27
queryCoeff	28
sc.cluster.trend	29
sc.filter	31
sc.p.vector	33
sc.restruct	36
sc.set.poly	37
sc.squeeze	39
sc.t.fit	41
ScMaSigPro	43
scmp.ob	45
scmpResults	46
showCoeff	47
showPoly	48
Significant	49
splat.sim	50
updateObject,ScMaSigPro-method	51

<i>as_scmp</i>	3
VariableProfiles	53
Index	54

<i>as_scmp</i>	<i>Create a ScMaSigPro object from a SingleCellExperiment</i>
----------------	---

Description

`as_scmp()` creates a [ScMaSigPro](#) object from a `SingleCellExperiment` whose cell metadata (`colData`) holds a pseudotime column and a path column. To start from a count matrix and a cell table, use `create_scmp()`; `assign_paths()` and `paths_from_graph()` build the two columns from the output of a trajectory tool.

Usage

```
as_scmp(  
  object,  
  from = "sce",  
  path_prefix = "Path",  
  root_label = "root",  
  ptime_col = "Pseudotime",  
  path_col = path_prefix,  
  verbose = TRUE,  
  additional_params = list(labels_exist = FALSE)  
)
```

Arguments

<code>object</code>	A <code>SingleCellExperiment</code> with the counts of the cells.
<code>from</code>	The class of object; only "sce" is supported.
<code>path_prefix</code>	The default of <code>path_col</code> , kept in the parameters.
<code>root_label</code>	The label of the root cells, kept in the parameters.
<code>ptime_col</code>	The name of the column of the cell metadata that holds the pseudotime of each cell.
<code>path_col</code>	The name of the column of the cell or bin metadata that holds the path label of each cell or bin.
<code>verbose</code>	Whether to print progress messages.
<code>additional_params</code>	A list: with <code>labels_exist = TRUE</code> , the columns <code>exist_ptime_col</code> and <code>exist_path_col</code> of the cell metadata are renamed to <code>ptime_col</code> and <code>path_col</code> .

Value

A [ScMaSigPro](#) object with the cells in the Sparse slot.

Author(s)

Priyansh Srivastava <spriyansh29@gmail.com>

See Also

Other input helpers: [assign_paths\(\)](#), [create_scmp\(\)](#), [paths_from_graph\(\)](#)

Examples

```
data("splat.sim", package = "scMaSigPro")
scmp <- as_scmp(splat.sim,
  additional_params = list(
    labels_exist = TRUE, exist_ptime_col = "Step",
    exist_path_col = "Group"
  ),
  verbose = FALSE
)
scmp
```

assign_paths

Assign cells to paths from lineage weights

Description

`assign_paths()` turns the lineage weights and pseudotimes of a trajectory tool, one column per lineage as `slingshot` and `tradeSeq` give them, into what `scMaSigPro` needs: one pseudotime value and one path label per cell. Cells on every lineage are labelled as the root, and cells on some of the lineages get the names of those lineages joined by `shared_sep`: the root and link paths that [sc.reconstruct\(\)](#) takes.

Usage

```
assign_paths(
  pseudotime,
  weights,
  min_weight = 0.5,
  root_label = "root",
  shared_sep = "_links_",
  verbose = TRUE
)
```

Arguments

pseudotime	The pseudotime of each cell: a numeric matrix with the row and column names of weights, such as <code>slingshot::slingPseudotime()</code> ; or, for a pseudotime that every lineage shares, as <code>Monocle3</code> gives it, a numeric vector with one value per row of weights, in that order or named by cell.
------------	--

weights	A matrix or data frame of non-negative weights with one row per cell, named by cell, and one column per lineage, named by lineage and at least two of them, such as <code>slingshot::slingCurveWeights()</code> , the cell weights of <code>tradeSeq</code> or the result of <code>paths_from_graph()</code> .
min_weight	A single positive number: a cell is on a lineage when its weight there is at least <code>min_weight</code> and its pseudotime there is a finite number. (Default is 0.5)
root_label	The path label of the cells on every lineage. (Default is "root")
shared_sep	The separator of the lineage names in the label of the cells on some but not all lineages; pass it to <code>sc.reconstruct()</code> as <code>link_sep</code> . (Default is "links", the default of <code>sc.reconstruct()</code>)
verbose	Print a message when cells are left out. (Default is TRUE)

Details

Each cell is placed on the lineages where its weight is at least `min_weight` and its pseudotime is known. A cell on one lineage gets the name of that lineage as its path, a cell on every lineage gets `root_label`, and a cell on two or more lineages but not all of them gets the names of those lineages, in the order of the columns of weights, joined by `shared_sep`. A cell on no lineage is left out. The pseudotime of a cell is the mean of its pseudotimes on its lineages, or its value when `pseudotime` is a vector.

The weights are compared with `min_weight` as they are given. Curve weights of `slingshot` are close to 1 on every lineage for the cells of the root; for weights that sum to 1 in each cell, as with `slingCurveWeights(as.probs = TRUE)`, use a `min_weight` below $1 / (\text{number of lineages})$ or the root cells are left out. Soft weights can place a few cells on a set of lineages that share no segment of the trajectory beyond the root, such as two of three lineages that split at the root; their label is built the same way, so relabel or remove them before `sc.reconstruct()` if they are not a link.

Value

A data frame with one row per cell that is on at least one lineage, in the order of the rows of `weights` and named by cell, and two columns: `Pseudotime` (numeric) and `Path` (character). Pass it to `create_scmp()` as `cell_data` with `ptime_col = "Pseudotime"` and `path_col = "Path"`, or add its columns to the `colData()` of a `SingleCellExperiment` with the same cells for `as_scmp()`, whose default column names they are.

Input contract

`scMaSigPro` needs one pseudotime value and one path label per cell, and nothing else from the tool that inferred the trajectory. `create_scmp()` takes them as a data frame named by cell, such as the result of `assign_paths()`, and `as_scmp()` as two `colData()` columns of a `SingleCellExperiment`. Code outside this package that reads a trajectory, such as a companion package for `Monocle3` objects or an interactive branch selector, returns that data frame, or the R code that builds it with `paths_from_graph()` and `assign_paths()`. Its path labels follow the rules in `Details`, so that `sc.reconstruct()` can rebuild the paths that share a root.

Author(s)

Priyansh Srivastava <spriyansh29@gmail.com>

References

Street K, Risso D, Fletcher RB, Das D, Ngai J, Yosef N, Purdom E, Dudoit S (2018). Slingshot: cell lineage and pseudotime inference for single-cell transcriptomics. BMC Genomics 19, 477. doi:10.1186/s1286401847720

Van den Berge K, Roux de Bézieux H, Street K, Saelens W, Cannoodt R, Saeys Y, Dudoit S, Clement L (2020). Trajectory-based differential expression analysis for single-cell sequencing data. Nature Communications 11, 1201. doi:10.1038/s41467020147663

See Also

[paths_from_graph\(\)](#) for weights from a trajectory graph; [sc.restruct\(\)](#) for the root and link paths.

Other input helpers: [as_scmp\(\)](#), [create_scmp\(\)](#), [paths_from_graph\(\)](#)

Examples

```
# Three lineages: cells 1 and 2 are on all of them, cell 3 on A and B
weights <- matrix(
  c(
    1, 1, 1, 1, 0, 0,
    1, 1, 1, 0, 1, 0,
    1, 0.9, 0.2, 0, 0, 1
  ),
  ncol = 3,
  dimnames = list(paste0("cell", 1:6), c("A", "B", "C"))
)
pseudotime <- c(0.1, 0.5, 1.2, 2.1, 2.3, 1.8)
assign_paths(pseudotime, weights)

# A pseudotime for each lineage, as slingshot gives it
pseudotime <- cbind(A = pseudotime, B = pseudotime + 0.1, C = pseudotime)
rownames(pseudotime) <- rownames(weights)
assign_paths(pseudotime, weights, shared_sep = "|")
```

cDense

Bin metadata of the Dense slot

Description

cDense() returns the bin metadata (colData) of the Dense slot of a [ScMaSigPro](#) object, one row per pseudotime bin; cDense<- replaces it.

Usage

```
cDense(object, value = "missing")

cDense(object) <- value

## S4 method for signature 'ScMaSigPro'
cDense(object, value = "missing")

## S4 replacement method for signature 'ScMaSigPro'
cDense(object) <- value
```

Arguments

object	A ScMaSigPro object.
value	A data frame of bin metadata, one row per bin.

Details

The bins are the columns of the pseudo-bulk counts. A value with one row per bin replaces the metadata and keeps the counts; its rows may be renamed, and [pb_counts\(\)](#) then recomputes the counts. A value with another number of rows, such as bins rebuilt by hand, cannot keep the counts: the Dense slot then holds the metadata without counts, a message says so, and [pb_counts\(\)](#) must be run before the next step. `cDense(object, value)` sets value as `cDense(object) <- value` does.

Value

`cDense(object)`: a data frame with one row per bin. `cDense<-` and `cDense(object, value)`: the updated object, returned invisibly by the latter.

See Also

Other accessors: [cSparse\(\)](#), [eDense\(\)](#), [eSparse\(\)](#), [pathAssign\(\)](#), [predictors\(\)](#)

Examples

```
data("scmp.ob", package = "scMaSigPro")
bins <- cDense(scmp.ob)
head(bins[, c("Path", "scmp_binned_pseudotime", "scmp_bin_size")])

# Add a column to the bin metadata
bins$early <- bins$scmp_binned_pseudotime <= 4
cDense(scmp.ob) <- bins
```

create_scmp

Create a ScMaSigPro object from counts and cell metadata

Description

create_scmp() creates a [ScMaSigPro](#) object from a count matrix and a table of cell metadata with the pseudotime and the path of each cell, such as the result of [assign_paths\(\)](#). Bins made elsewhere can be given too.

Usage

```
create_scmp(
  counts,
  cell_data,
  feature_data = NULL,
  bin_counts = NULL,
  bin_cell_data = NULL,
  ptime_col,
  path_col,
  use_as_bin = FALSE
)
```

Arguments

counts	A matrix of raw counts, genes by cells.
cell_data	A data frame of cell metadata, one row per column of counts, named like them.
feature_data	An optional data frame of gene metadata, one row per gene of counts, named like them or with automatic row names. It becomes the rowData() of the Sparse slot, and of the Dense slot when this function builds it.
bin_counts	Optional pseudo-bulk counts of bins made outside scMaSigPro: a matrix with the genes of counts as rows, in the same order, and one column per bin. Needs bin_cell_data; not with use_as_bin = TRUE.
bin_cell_data	A data frame with one row per bin, named like the columns of bin_counts, with the binned pseudotime in ptime_col and the path in path_col. A scmp_bin_size column gives the number of cells of each bin; without it the sizes are NA.
ptime_col	The name of the column of the cell metadata that holds the pseudotime of each cell.
path_col	The name of the column of the cell or bin metadata that holds the path label of each cell or bin.
use_as_bin	Whether every cell is a bin of its own, without pseudo-bulking, as for bulk time-course data.

Details

By default the Dense slot is empty and `sc.squeeze()` bins the cells. With `use_as_bin = TRUE` every cell is a bin. With `bin_counts` and `bin_cell_data` it holds bins made outside `scMaSigPro` and the workflow continues with `sc.set.poly()`; such bins have no member cells, so `pb_counts()` and `sc.restruct()` do not apply.

Value

A `ScMaSigPro` object with the cells in the Sparse slot and, with `use_as_bin = TRUE` or `bin_counts`, the bins in the Dense slot.

Author(s)

Priyansh Srivastava <spriyansh29@gmail.com>

See Also

Other input helpers: `as_scmp()`, `assign_paths()`, `paths_from_graph()`

Examples

```
data("splat.sim", package = "scMaSigPro")
counts <- SummarizedExperiment::assay(splat.sim, "counts")
cells <- as.data.frame(SummarizedExperiment::colData(splat.sim))
scmp <- create_scmp(counts, cells, ptime_col = "Step", path_col = "Group")
scmp
```

cSparse

Cell metadata of the Sparse slot

Description

`cSparse()` returns the cell metadata (`colData`) of the Sparse slot of a `ScMaSigPro` object; `cSparse<-` replaces it.

Usage

```
cSparse(object, value = "missing")

cSparse(object) <- value

## S4 method for signature 'ScMaSigPro'
cSparse(object, value = "missing")

## S4 replacement method for signature 'ScMaSigPro'
cSparse(object) <- value
```

Arguments

object	A ScMaSigPro object.
value	A data frame with one row per cell, named after the cells of the Sparse slot and in their order, as <code>cSparse()</code> returns it. Without row names, the rows are taken in the order of the cells.

Details

The cells are the columns of the counts, and [sc.squeeze\(\)](#) finds them by name, so a value with other cells or another order is an error. `cSparse(object, value)` sets value as `cSparse(object) <- value` does.

Value

`cSparse(object)`: a data frame with one row per cell. `cSparse<-` and `cSparse(object, value)`: the updated object, returned invisibly by the latter.

See Also

Other accessors: [cDense\(\)](#), [eDense\(\)](#), [eSparse\(\)](#), [pathAssign\(\)](#), [predictors\(\)](#)

Examples

```
data("scmp.ob", package = "scMaSigPro")
cells <- cSparse(scmp.ob)
head(cells[, c("Pseudotime", "Path")])

# Add a column to the cell metadata
cells$early <- cells$Pseudotime < 50
cSparse(scmp.ob) <- cells
```

eDense

Pseudo-bulk counts of the Dense slot

Description

`eDense()` returns an assay of the Dense slot of a [ScMaSigPro](#) object, genes by bins, as a dense matrix; `eDense<-` sets one.

Usage

```
eDense(object, i = "bulk.counts", ...)

eDense(object, i = "bulk.counts") <- value

## S4 method for signature 'ScMaSigPro'
eDense(object, i = "bulk.counts", ...)
```

```
## S4 replacement method for signature 'ScMaSigPro,character'
eDense(object, i = "bulk.counts") <- value

## S4 replacement method for signature 'ScMaSigPro,missing'
eDense(object, i = "bulk.counts") <- value
```

Arguments

object	A ScMaSigPro object.
i	The name of the assay. With the default, "bulk.counts", eDense() returns the first assay when no assay has that name.
...	Not used.
value	A matrix, a dgCMatrix or a data frame of numbers, with one row per gene and one column per bin of the Dense slot. Its row and column names, when present, must be the gene and bin names of the slot in the same order; missing names are filled in.

Details

eDense<- stores value as a dgCMatrix. It gives a message when it creates a new assay and a warning when it replaces one.

Value

eDense(): a numeric matrix. eDense<-: the updated object.

See Also

Other accessors: [cDense\(\)](#), [cSparse\(\)](#), [eSparse\(\)](#), [pathAssign\(\)](#), [predictors\(\)](#)

Examples

```
data("scmp.ob", package = "scMaSigPro")
bulk <- eDense(scmp.ob)
bulk[1:3, 1:4]

# Add a second assay
eDense(scmp.ob, "logcounts") <- log1p(bulk)
```

eSparse

Cell counts of the Sparse slot

Description

eSparse() returns an assay of the Sparse slot of a [ScMaSigPro](#) object, genes by cells, as a dense matrix; eSparse<- sets one.

Usage

```
eSparse(object, i = "counts", ...)

eSparse(object, i = "counts") <- value

## S4 method for signature 'ScMaSigPro'
eSparse(object, i = "counts", ...)

## S4 replacement method for signature 'ScMaSigPro,character'
eSparse(object, i = "counts") <- value

## S4 replacement method for signature 'ScMaSigPro,missing'
eSparse(object, i = "counts") <- value
```

Arguments

object	A ScMaSigPro object.
i	The name of the assay. With the default, "counts", eSparse() returns the first assay when no assay has that name.
...	Not used.
value	A matrix, a dgCMatrix or a data frame of numbers, with one row per gene and one column per cell of the Sparse slot. Its row and column names, when present, must be the gene and cell names of the slot in the same order; missing names are filled in.

Details

eSparse<- stores value as a dgCMatrix. It gives a message when it creates a new assay and a warning when it replaces one.

Value

eSparse(): a numeric matrix. eSparse<-: the updated object.

See Also

Other accessors: [cDense\(\)](#), [cSparse\(\)](#), [eDense\(\)](#), [pathAssign\(\)](#), [predictors\(\)](#)

Examples

```
data("scmp.ob", package = "scMaSigPro")
counts <- eSparse(scmp.ob)
counts[1:3, 1:4]

# Add a second assay
eSparse(scmp.ob, "logcounts") <- log1p(counts)
```

Estimates

Estimates

Description

The models that `sc.t.fit()` selected.

Details

`validObject()` checks that `t_score_matrix` has the shape and rows of `coefficient_matrix`, that `path_coefficient_matrix` has its rows, and that every row of `significance_matrix` is a row of `coefficient_matrix`.

Value

An object of class `Estimates`: the selected model of each gene, which `sc.t.fit()` stores in the `Estimate` slot of a `ScMaSigPro` object. `showSol()` and the other result readers return its matrices.

Slots

`significance_matrix` For each gene with a significant selected model, its p-value and R-squared and the p-value of each term (`showSol()`).

`coefficient_matrix` The coefficients (`showCoeff()`).

`path_coefficient_matrix` The coefficients of the polynomial of each path (`showGroupCoeff()`).

`t_score_matrix` The t statistics (`showTS()`).

`path` The path of the intercept and of each term.

`influential` The genes with influential observations (`showInflu()`).

See Also

Other classes: `MatrixDesign`, `ParameterConfig`, `ScMaSigPro`, `Significant`, `VariableProfiles`, `updateObject`, `ScMaSigPro`-method

Examples

```
data("scmp.ob", package = "scMaSigPro")
head(showCoeff(scmp.ob))
```

MatrixDesign

MatrixDesign

Description

The design of the polynomial model of `sc.set.poly()`.

Details

`validObject()` checks that `assignment_matrix` has the rows of `predictor_matrix` (one per bin), that `offset` is empty or has one value per row and that `groups.vector` has one value per column of `predictor_matrix`.

Value

An object of class `MatrixDesign`: the design of the polynomial model, which `sc.set.poly()` stores in the `Design` slot of a `ScMaSigPro` object, with the offsets that `sc.p.vector()` adds. `predictors()` and `pathAssign()` return its matrices.

Slots

`predictor_matrix` The design matrix: one row per bin, one column per term (`predictors()`).

`groups.vector` The path of each column of `predictor_matrix`: the reference path for its powers of pseudotime, `<path>vs<ref>` for the terms of another path.

`assignment_matrix` The path assignment matrix: one row per bin, with the binned pseudotime, a replicate index and a 0/1 column per path (`pathAssign()`).

`offset` The offset of each bin, the logarithm of its size factor (see `sc.p.vector()`), or an empty vector when the fits used no offset or no fit was made yet.

See Also

Other classes: `Estimates`, `ParameterConfig`, `ScMaSigPro`, `Significant`, `VariableProfiles`, `updateObject`, `ScMaSigPro-method`

Examples

```
data("scmp.ob", package = "scMaSigPro")
head(predictors(scmp.ob))
```

multi.lin.sce

*Simulated multifurcating trajectory***Description**

A SingleCellExperiment with the counts of 501 genes in 1977 cells of a simulated trajectory with three end points and a shared root, with the pseudotime and the path labels of each cell. The vignettes [Trajectory input](#) and [Shared root](#) analyse it.

Usage

```
data("multi.lin.sce", package = "scMaSigPro")
```

Format

A SingleCellExperiment with 501 genes (rows) and 1977 cells (columns):

counts **assay** The simulated counts.

colData cell_id, the cell name; group, the path label: "root" for the cells on all three lineages, "path_Y_15", "path_Y_18" or "path_Y_52" for the cells on one, and "path_Y_18|path_Y_15" for the cells on these two; Monocle3_Pseudotime, the pseudotime of Monocle 3.

reducedDims TSNE, the two dimensions of the t-SNE embedding.

The rowData has no columns.

Author(s)

Priyansh Srivastava <spriyansh29@gmail.com>

Source

The multifurcating simulation of Van den Berge et al. (2020), made with dyntoy (Saelens et al. 2019): multifurcating_4.rds of <https://github.com/statOmics/tradeSeqPaper>. Monocle 3 (Cao et al. 2019) normalised the counts, embedded the cells with PCA and t-SNE, learnt the principal graph and gave the pseudotime from the root node Y_51. Each cell is on the lineages whose shortest path from the root, on that graph, holds its nearest node, as in the Monocle 3 steps of the tradeSeq vignette; three lineages are kept and the cells on none of them are left out. The script is system.file("scripts", "make-multi-lin-sce.R", package = "scMaSigPro").

References

- Van den Berge K, Roux de Bézieux H, Street K, Saelens W, Cannoodt R, Saeys Y, Dudoit S, Clement L (2020). Trajectory-based differential expression analysis for single-cell sequencing data. Nature Communications 11, 1201. doi:10.1038/s41467020147663
- Saelens W, Cannoodt R, Todorov H, Saeys Y (2019). A comparison of single-cell trajectory inference methods. Nature Biotechnology 37(5), 547-554. doi:10.1038/s4158701900719
- Cao J, Spielmann M, Qiu X, Huang X, Ibrahim DM, Hill AJ, Zhang F, Mundlos S, Christiansen L, Steemers FJ, Trapnell C, Shendure J (2019). The single-cell transcriptional landscape of mammalian organogenesis. Nature 566(7745), 496-502. doi:10.1038/s415860190969x

See Also

Other datasets: [scmp.ob](#), [splat.sim](#)

Examples

```
data("multi.lin.sce", package = "scMaSigPro")
table(SummarizedExperiment::colData(multi.lin.sce)$group)
```

ParameterConfig

ParameterConfig

Description

The settings of an analysis: the names of the columns of the cell and bin metadata and the parameters of every step of the workflow.

Details

`validObject()` checks that every slot except `distribution` holds a single non-missing value, that `min_counts` and `min_bins_expressed` are at least 0, that `test` is "chisq" or "F" and that `distribution` is a `stats::family()` object.

Value

An object of class `ParameterConfig`: the settings of an analysis, one value per slot. Every step of the workflow records the settings it used in the `Parameters` slot of a `ScMaSigPro` object, and `showParams()` lists them.

Slots

`bin_ptime_col` The column of the bin metadata with the binned pseudotime.

`path_prefix` The default path column of `as_scmp()`.

`root_label` The label of the root cells given to `as_scmp()`, kept for reference.

`ptime_col` The column of the cell metadata with the pseudotime.

`bin_method` The rule for the number of bins of `sc.squeeze()`.

`path_col` The column of the cell and bin metadata with the path labels.

`bin_col` The column of the bin metadata with the bin names.

`bin_size_col` The column of the bin metadata with the number of cells of each bin.

`bin_mem_col` The column of the bin metadata with the cells of each bin.

`anno_col` The column of the cell metadata with cell annotations; not used by the steps.

`g` The number of genes that `sc.p.vector()` fitted: those left after removing the genes with fewer than `min_na` values, with zero counts in every bin, or below `min_counts` or `min_bins_expressed`.

`p_value` The level of `sc.p.vector()` and `sc.t.fit()`.

`min_na` The minimum number of bins with a value (not NA) that `sc.p.vector()` required; it only removes genes from counts with missing values.

`mt_correction` The multiple-testing correction of `sc.p.vector()`.

`link` The link function of the fits.

`epsilon` The convergence tolerance of the fits.

`selection_method` The stepwise method of `sc.t.fit()`.

`offset` Whether the fits used the offset of each bin, the logarithm of its size factor (see `sc.p.vector()`).

`max_it` The maximum number of iterations of the fits.

`poly_degree` The degree of the polynomial of `sc.set.poly()`.

`distribution` The family of the fits, a `stats::family()` object; with an estimated theta it also holds `theta_estimate` and `theta_gene` (see `sc.p.vector()`).

`cluster_method` The clustering method of `sc.cluster.trend()`.

`use_dim` How `sc.cluster.trend()` filled missing values: by column or by row.

`fill_na` What `sc.cluster.trend()` filled missing values with.

`min_counts` The minimum total count over the bins that `sc.p.vector()` required of a gene (0: no minimum).

`min_bins_expressed` The minimum number of bins with a count above 0 that `sc.p.vector()` required of a gene (0: no minimum).

`test` The test of the full model of each gene that `sc.p.vector()` was asked for: "chisq", the chi-squared test, or "F", the F test.

See Also

Other classes: `Estimates`, `MatrixDesign`, `ScMaSigPro`, `Significant`, `VariableProfiles`, `updateObject`, `ScMaSigPro-m`

Examples

```
data("scmp.ob", package = "scMaSigPro")
showParams(scmp.ob)
```

pathAssign

Path assignment matrix of the design

Description

`pathAssign()` returns the path assignment matrix that `sc.set.poly()` stores in the `Design` slot of a `ScMaSigPro` object: one row per bin, with the binned pseudotime, a replicate index and a 0/1 column per path. `pathAssign<-` replaces it.

Usage

```

pathAssign(object, value = "missing")

pathAssign(object) <- value

## S4 method for signature 'ScMaSigPro'
pathAssign(object, value = "missing")

## S4 replacement method for signature 'ScMaSigPro'
pathAssign(object) <- value

```

Arguments

object	A ScMaSigPro object.
value	A matrix with the rows of the design matrix (predictors()), one per bin and with the same names.

Details

pathAssign(object, value) sets value as pathAssign(object) <- value does.

Value

pathAssign(object): a numeric matrix. pathAssign<- and pathAssign(object, value): the updated object, returned invisibly by the latter.

See Also

Other accessors: [cDense\(\)](#), [cSparse\(\)](#), [eDense\(\)](#), [eSparse\(\)](#), [predictors\(\)](#)

Examples

```

data("scmp.ob", package = "scMaSigPro")
head(pathAssign(scmp.ob))

```

paths_from_graph

Lineage weights from a trajectory graph

Description

paths_from_graph() gives the lineage weights of the cells of a trajectory graph, such as the principal graph of Monocle3 or a minimum spanning tree of clusters, for [assign_paths\(\)](#). Each lineage is the shortest path from the root node to an end node, and a cell is on it (weight 1) when its node is on that path.

Usage

```
paths_from_graph(cell_node, edges, root, endpoints = NULL)
```

Arguments

cell_node	A vector with the node of each cell, named by cell. The nodes are compared with the node names of edges as character strings.
edges	A matrix or data frame whose first two columns hold the two nodes of each edge of the graph, which is taken as undirected.
root	The name of the root node.
endpoints	The names of the end nodes, one lineage each; the default, NULL, takes every node with one edge other than the root, in the order in which they first appear in edges.

Details

The paths are found with `igraph::shortest_paths()` on the unweighted graph, so this function needs the suggested package **igraph**. In a tree, as principal graphs and minimum spanning trees are, the path from the root to each end node is the only one. Every node of `cell_node` must be a node of edges, and every end node must be reachable from the root.

Value

A numeric matrix with one row per cell, in the order and with the names of `cell_node`, and one column per end node, named by it, holding 1 when the node of the cell is on the path from the root to that end node and 0 otherwise.

Author(s)

Priyansh Srivastava <spriyansh29@gmail.com>

References

Van den Berge K, Roux de Bézieux H, Street K, Saelens W, Cannoodt R, Saeys Y, Dudoit S, Clement L (2020). Trajectory-based differential expression analysis for single-cell sequencing data. *Nature Communications* 11, 1201. doi:[10.1038/s41467020147663](https://doi.org/10.1038/s41467020147663)

Csárdi G, Nepusz T (2006). The igraph software package for complex network research. *InterJournal Complex Systems*, 1695. <https://igraph.org>

See Also

[assign_paths\(\)](#), which turns these weights into path labels.

Other input helpers: [as_scmp\(\)](#), [assign_paths\(\)](#), [create_scmp\(\)](#)

Examples

```
# A tree: the root R, a segment A-B shared by the end nodes C and D, and
# the end node E that branches off at A
edges <- rbind(
  c("R", "A"), c("A", "B"), c("B", "C"), c("B", "D"), c("A", "E")
)
cell_node <- c(
  cell1 = "R", cell2 = "A", cell3 = "B", cell4 = "C", cell5 = "D",
  cell6 = "E"
)
if (requireNamespace("igraph", quietly = TRUE)) {
  weights <- paths_from_graph(cell_node, edges, root = "R")
  print(weights)
  # With a pseudotime that every lineage shares
  assign_paths(c(0, 1, 2, 3, 3, 2), weights)
}
```

pb_counts

Pseudo-bulk counts of the bins

Description

pb_counts() adds up, or averages, the counts of the cells of each bin into the pseudo-bulk counts (bulk.counts) of the Dense slot. [sc.squeeze\(\)](#) and [sc.restruct\(\)](#) call it; call it after changing the bins with [cDense\(\)](#).

Usage

```
pb_counts(
  scmpObj,
  bin_mem_col = scmpObj@Parameters@bin_mem_col,
  bin_col = scmpObj@Parameters@bin_col,
  assay_name = "counts",
  cluster_count_by = "sum"
)
```

Arguments

scmpObj	A ScMaSigPro object.
bin_mem_col	The name of the column of the bin metadata that holds the cells of each bin, separated by .
bin_col	The name of the column of the bin metadata that holds the name of each bin.
assay_name	The name of the assay of the Sparse slot that holds the counts of the cells.
cluster_count_by	How the counts of the cells of a bin are combined: "sum" or "mean" (rounded to whole numbers).

Details

The bins are the rows of the bin metadata (`cDense()`): their row names, or the `bin_col` column when there are none, name the columns of the counts, and `bin_mem_col` gives their cells.

Value

The object whose `Dense` slot holds the pseudo-bulk counts (`bulk.counts`, genes by bins) and the bin metadata.

Author(s)

Priyansh Srivastava <spriyansh29@gmail.com>

References

Srivastava P, Benegas Coll M, Götz S, Nueda MJ, Conesa A (2024). scMaSigPro: differential expression analysis along single-cell trajectories. *Bioinformatics* 40(7), btae443. doi:10.1093/bioinformatics/btae443

See Also

Other binning helpers: `calc_bin_size()`, `estBinSize()`, `extract_interval()`, `optimize_bin_max()`

Examples

```
data("scmp.ob", package = "scMaSigPro")
# The mean instead of the sum of the counts of the cells of each bin
scmp <- pb_counts(scmp.ob, cluster_count_by = "mean")
eDense(scmp)[1:3, 1:4]
```

plotBinTile

Plot the sizes of the bins

Description

`plotBinTile()` draws a tile for each bin, by path and binned pseudotime, coloured and labelled by its number of cells.

Usage

```
plotBinTile(
  scmpObj,
  path_col = scmpObj@Parameters@path_col,
  bin_size_col = scmpObj@Parameters@bin_size_col,
  bin_ptime_col = scmpObj@Parameters@bin_ptime_col
)
```

Arguments

scmpObj	A ScMaSigPro object.
path_col	The name of the column of the cell or bin metadata that holds the path label of each cell or bin.
bin_size_col	The name of the column of the bin metadata that holds the number of cells of each bin.
bin_ptime_col	The name of the column of the bin metadata that holds the binned pseudotime of each bin.

Value

A ggplot object.

Author(s)

Priyansh Srivastava <spriyansh29@gmail.com>

See Also

Other plots: [plotDiagnostics\(\)](#), [plotIntersect\(\)](#), [plotTrend\(\)](#), [plotTrendCluster\(\)](#)

Examples

```
data("scmp.ob", package = "scMaSigPro")
plotBinTile(scmp.ob)
```

plotDiagnostics	<i>Plot the diagnostics of the model of a gene</i>
-----------------	--

Description

`plotDiagnostics()` draws four diagnostic plots of a model of a gene of the gene lists of [sc.filter\(\)](#): the normal Q-Q plot of the standardised residuals, the standardised residuals and their square root against the fitted values (scale-location), and the observed against the predicted counts.

Usage

```
plotDiagnostics(scmpObj, feature_id, model = "optimized")
```

Arguments

scmpObj	A ScMaSigPro object.
feature_id	The name of a gene.
model	The model: "optimized", the model selected by sc.t.fit() ; "full"; or "intercept", the intercept-only model.

Value

A patchwork object with four ggplot panels.

Author(s)

Priyansh Srivastava <spriyansh29@gmail.com>

See Also

Other plots: [plotBinTile\(\)](#), [plotIntersect\(\)](#), [plotTrend\(\)](#), [plotTrendCluster\(\)](#)

Examples

```
data("scmp.ob", package = "scMaSigPro")
if (requireNamespace("patchwork", quietly = TRUE)) {
  plotDiagnostics(scmp.ob, "Gene10")
}
```

plotIntersect

Plot the overlaps of the gene lists

Description

plotIntersect() draws an UpSet plot of the gene lists of [sc.filter\(\)](#) with UpSetR::upset(), which needs the suggested package UpSetR, or returns its data.

Usage

```
plotIntersect(
  scmpObj,
  min_intersection_size = 1,
  keep_empty_groups = FALSE,
  return = FALSE
)
```

Arguments

scmpObj	A ScMaSigPro object.
min_intersection_size	The smallest intersection drawn.
keep_empty_groups	Whether to draw the empty intersections.
return	Whether to return the data instead of the plot.

Value

An upset object of UpSetR or, with return = TRUE, a data frame with one row per gene and a 0/1 column per gene list.

Author(s)

Priyansh Srivastava <spriyansh29@gmail.com>

References

Conway JR, Lex A, Gehlenborg N (2017). UpSetR: an R package for the visualization of intersecting sets and their properties. *Bioinformatics* 33(18), 2938-2940. doi:10.1093/bioinformatics/btx364

See Also

Other plots: [plotBinTile\(\)](#), [plotDiagnostics\(\)](#), [plotTrend\(\)](#), [plotTrendCluster\(\)](#)

Examples

```
data("scmp.ob", package = "scMaSigPro")
# The genes of each list as 0/1 columns
head(plotIntersect(scmp.ob, return = TRUE))
if (requireNamespace("UpSetR", quietly = TRUE)) {
  plotIntersect(scmp.ob)
}
```

plotTrend

Plot the trend of a gene

Description

plotTrend() plots the pseudo-bulk counts of a gene along the binned pseudotime of each path, with the curves of its selected model ([sc.t.fit\(\)](#)).

Usage

```
plotTrend(
  scmpObj,
  feature_id,
  xlab = "Pooled Pseudotime",
  ylab = "Pseudobulk Expression",
  smoothness = 0.01,
  logs = NULL,
  logType = "log",
  pseudoCount = 1,
  significant = TRUE,
```

```

summary_mode = "median",
curves = TRUE,
lines = FALSE,
points = TRUE
)

```

Arguments

scmpObj	A ScMaSigPro object.
feature_id	The name of a gene.
xlab	The label of the x axis.
ylab	The label of the y axis; with logged points it becomes, for example, log(ylab).
smoothness	The step of the pseudotime grid of the curves: the smaller, the smoother.
logs	Whether to log the points: the counts of the bins, divided, for log-link fits, by the size factors stored in the design. The fitted curve is the polynomial of the selected model on the scale of the link: the log of the mean under the log link, the mean itself under the identity link. The default, NULL, logs the points when the fits used the log link and draws them as they are otherwise, so that points and curve are on the same scale.
logType	The log of the points: "log", "log2" or "log10".
pseudoCount	The number added to the points before the log.
significant	Whether the gene must be in a gene list of sc.filter() .
summary_mode	How the points of each binned pseudotime and path are summarised for the lines: "median" or "mean".
curves	Whether to draw the fitted curves.
lines	Whether to draw lines through the points.
points	Whether to draw the points.

Value

A ggplot object.

Author(s)

Priyansh Srivastava <spriyansh29@gmail.com>

See Also

Other plots: [plotBinTile\(\)](#), [plotDiagnostics\(\)](#), [plotIntersect\(\)](#), [plotTrendCluster\(\)](#)

Examples

```

data("scmp.ob", package = "scMaSigPro")
plotTrend(scmp.ob, "Gene10")

```

plotTrendCluster	<i>Plot the trends of the clusters</i>
------------------	--

Description

plotTrendCluster() plots, in one panel per cluster of [sc.cluster.trend\(\)](#), the points of its genes along the binned pseudotime and a loess curve per path through their fitted curves.

Usage

```
plotTrendCluster(
  scmpObj,
  xlab = "Pooled Pseudotime",
  ylab = NULL,
  summary_mode = "median",
  logs = NULL,
  logType = "log",
  smoothness = 1,
  verbose = TRUE,
  pseudoCount = 1,
  significant = FALSE,
  curves = TRUE,
  lines = FALSE,
  points = TRUE,
  loess_span = 0.8
)
```

Arguments

scmpObj	A ScMaSigPro object.
xlab	The label of the x axis.
ylab	The label of the y axis. The default, NULL, is "log(Pseudobulk Expression)" for log-link fits and when the points are logged, and "Pseudobulk Expression" otherwise.
summary_mode	How the points of each binned pseudotime and path are summarised for the lines: "median" or "mean".
logs	Whether to log the points, as in plotTrend() . The default, NULL, logs them when the fits used the log link.
logType	The log of the points: "log", "log2" or "log10".
smoothness	The step of the pseudotime grid of the curves: the smaller, the smoother.
verbose	Whether to print progress messages.
pseudoCount	The number added to the points before the log.
significant	Whether the genes must be in a gene list of sc.filter() ; the others are left out with a warning.

curves	Whether to draw the fitted curves.
lines	Whether to draw lines through the points.
points	Whether to draw the points.
loess_span	The span of the loess curves.

Value

A ggplot object with one panel per cluster.

Author(s)

Priyansh Srivastava <spriyansh29@gmail.com>

See Also

Other plots: [plotBinTile\(\)](#), [plotDiagnostics\(\)](#), [plotIntersect\(\)](#), [plotTrend\(\)](#)

Examples

```
data("scmp.ob", package = "scMaSigPro")
plotTrendCluster(scmp.ob, verbose = FALSE)
```

predictors

Design matrix of the polynomial model

Description

`predictors()` returns the design matrix that [sc.set.poly\(\)](#) stores in the Design slot of a [ScMaSigPro](#) object: one row per bin and one column per term of the full model. `predictors<-` replaces it.

Usage

```
predictors(object, value = "missing")

predictors(object) <- value

## S4 method for signature 'ScMaSigPro'
predictors(object, value = "missing")

## S4 replacement method for signature 'ScMaSigPro'
predictors(object) <- value
```

Arguments

object A [ScMaSigPro](#) object.

value A matrix with the rows of the path assignment matrix ([pathAssign\(\)](#)), one per bin and with the same names, and one column per term of the design.

Details

`predictors(object, value)` sets `value` as `predictors(object) <- value` does.

Value

`predictors(object)`: a numeric matrix. `predictors<-` and `predictors(object, value)`: the updated object, returned invisibly by the latter.

See Also

Other accessors: [cDense\(\)](#), [cSparse\(\)](#), [eDense\(\)](#), [eSparse\(\)](#), [pathAssign\(\)](#)

Examples

```
data("scmp.ob", package = "scMaSigPro")
head(predictors(scmp.ob))
```

queryCoeff

Query the coefficients of the selected models

Description

`queryCoeff()` returns the coefficients of a group of terms for the genes whose selected model ([sc.t.fit\(\)](#)) passes `rsq` and `p_value` and has a non-zero coefficient for every term of the group. It also prints the formula of the model.

Usage

```
queryCoeff(
  scmpObj,
  rsq = 0.7,
  p_value = scmpObj@Parameters@p_value,
  query = "pseudotime_path",
  change = NULL,
  strictly = FALSE,
  verbose = TRUE
)
```

Arguments

scmpObj	A ScMaSigPro object.
rsq	The smallest R-squared of the selected model of a gene.
p_value	The largest p-value of the selected model of a gene; by default that of sc.p.vector() .
query	The terms: "pseudotime", the powers of pseudotime of the reference path; "pseudotime_path" or "path_pseudotime", their products with the path dummies; "path", the path dummies.
change	NULL, or "increasing" or "decreasing": keep the genes whose coefficients of the query add up to a positive (negative) number, or with strictly = TRUE are all positive (negative).
strictly	See change.
verbose	Whether to print progress messages.

Value

A data frame of coefficients: one row per gene that passes the thresholds and the query, one column per queried term.

Author(s)

Priyansh Srivastava <spriyansh29@gmail.com>.

See Also

[showCoeff\(\)](#) for the coefficients of every gene.

Other results: [scmpResults\(\)](#), [showCoeff\(\)](#), [showPoly\(\)](#)

Examples

```
data("scmp.ob", package = "scMaSigPro")
# Genes whose coefficients of pseudotime are all positive
queryCoeff(scmp.ob,
  query = "pseudotime", change = "increasing", strictly = TRUE,
  verbose = FALSE
)
```

sc.cluster.trend

Cluster the significant genes

Description

sc.cluster.trend() clusters the genes of the gene lists of [sc.filter\(\)](#) by the coefficients of their selected models or by their pseudo-bulk counts, as [see.genes\(\)](#) of [maSigPro](#) does, with hierarchical clustering, k-means or Gaussian mixture models (mclust).

Usage

```

sc.cluster.trend(
  scmpObj,
  geneSet = "intersect",
  cluster_by = "coeff",
  cluster_method = "hclust",
  hclust.agglo.method = "ward.D",
  kmeans.iter.max = 500,
  hclust.distance = "cor",
  includeInflu = TRUE,
  mclust.k = FALSE,
  k = 9,
  fill_na = "zero",
  use_dim = "col"
)

```

Arguments

scmpObj	A ScMaSigPro object.
geneSet	The genes to cluster: the name of a list of sc.filter() , "intersect", the genes in every list, or "union", the genes in any list.
cluster_by	What to cluster: the coefficients of the selected models ("coeff") or the pseudo-bulk counts ("counts").
cluster_method	"hclust", "kmeans" or "Mclust", which needs the suggested package mclust.
hclust.agglo.method	The agglomeration method of stats::hclust() .
kmeans.iter.max	The maximum number of iterations of stats::kmeans() .
hclust.distance	The distance of "hclust": "cor", one minus the correlation of two genes, or a method of stats::dist() .
includeInflu	Whether to keep the genes with influential observations (see showInflu()).
mclust.k	Whether "Mclust" chooses the number of clusters instead of taking k.
k	The number of clusters.
fill_na	How missing values are filled before clustering: with "zero", or the "mean" or "median" of their column or row (use_dim).
use_dim	Whether fill_na takes the mean or median by column ("col") or by row ("row").

Value

The object with the cluster of each gene in the Significant slot (class [Significant](#)).

Author(s)

Priyansh Srivastava <spriyansh29@gmail.com>, Ana Conesa and Maria Jose Nueda, <mj.nueda@ua.es>

References

Conesa A, Nueda MJ, Ferrer A, Talón M (2006). maSigPro: a method to identify significantly differential expression profiles in time-course microarray experiments. *Bioinformatics* 22(9), 1096-1102. doi:10.1093/bioinformatics/btl056

Scrucca L, Fop M, Murphy TB, Raftery AE (2016). mclust 5: clustering, classification and density estimation using Gaussian finite mixture models. *The R Journal* 8(1), 289-317. doi:10.32614/RJ-2016021

See Also

`plotTrendCluster()`, which plots the clusters.

Other workflow steps: `sc.filter()`, `sc.p.vector()`, `sc.restruct()`, `sc.set.poly()`, `sc.squeeze()`, `sc.t.fit()`

Examples

```
data("scmp.ob", package = "scMaSigPro")
# Cluster the pseudo-bulk counts of the genes of every list
scmp <- sc.cluster.trend(scmp.ob,
  geneSet = "union", cluster_by = "counts", k = 3
)
table(scmpResults(scmp)$cluster)
```

sc.filter	<i>List the significant genes</i>
-----------	-----------------------------------

Description

`sc.filter()` lists the genes whose selected model (`sc.t.fit()`) has an R-squared of at least `rsq` and a p-value of at most `p_value`: in one list, one list per term or one list per path (vars), as `get.siggenes()` of `maSigPro` does.

Usage

```
sc.filter(
  scmpObj,
  rsq = 0.7,
  p_value = scmpObj@Parameters@p_value,
  vars = c("all", "each", "groups"),
  intercept = "dummy",
  term_p_value = 0.05,
  includeInflu = TRUE
)
```

Arguments

scmpObj	A ScMaSigPro object.
rsq	The smallest R-squared of the selected model of a gene.
p_value	The largest p-value of the selected model of a gene; by default that of sc.p.vector() .
vars	"all": one list with every gene that passes; "each": one list per term, with the genes whose term passes <code>term_p_value</code> ; "groups": one list per path, with the genes with a term of that path that passes it.
intercept	With <code>vars = "groups"</code> , the intercept terms that count: "dummy", the dummies of the paths (the differences of their intercepts from the reference); "all", these and the intercept of the reference path; "none", neither, so that only the terms in pseudotime count.
term_p_value	The largest p-value of a term, for <code>vars = "each"</code> and <code>vars = "groups"</code> .
includeInflu	Whether to keep the genes with influential observations (see showInflu()).

Value

The object with the gene lists in the Significant slot (class [Significant](#)); earlier clusters are removed.

Author(s)

Priyansh Srivastava <spriyansh29@gmail.com>, Ana Conesa and Maria Jose Nueda, <mj.nueda@ua.es>

References

Conesa A, Nueda MJ, Ferrer A, Talón M (2006). maSigPro: a method to identify significantly differential expression profiles in time-course microarray experiments. *Bioinformatics* 22(9), 1096-1102. doi:10.1093/bioinformatics/btl056

Nueda MJ, Tarazona S, Conesa A (2014). Next maSigPro: updating maSigPro bioconductor package for RNA-seq time series. *Bioinformatics* 30(18), 2598-2602. doi:10.1093/bioinformatics/btu333

See Also

[plotIntersect\(\)](#) for the overlaps of the lists; [Significant](#).

Other workflow steps: [sc.cluster.trend\(\)](#), [sc.p.vector\(\)](#), [sc.restruct\(\)](#), [sc.set.poly\(\)](#), [sc.squeeze\(\)](#), [sc.t.fit\(\)](#)

Examples

```
data("scmp.ob", package = "scMaSigPro")
scmp <- sc.filter(scmp.ob, rsq = 0.6, vars = "groups")
res <- scmpResults(scmp)
# The number of genes in each list
c(Path1 = sum(res$sig_Path1), Path2vsPath1 = sum(res$sig_Path2vsPath1))
```

sc.p.vector

Test the full model of each gene

Description

sc.p.vector() fits the full polynomial model of each gene to its pseudo-bulk counts and tests it against the intercept-only model. The genes whose adjusted p-value is at most p_value have a non-flat profile; [sc.t.fit\(\)](#) selects the terms of their models.

Usage

```
sc.p.vector(
  scmpObj,
  p_value = 0.05,
  mt_correction = "BH",
  min_na = 6,
  min_counts = 0,
  min_bins_expressed = 0,
  family = negative.binomial(theta = 10, link = link),
  epsilon = 1e-08,
  verbose = TRUE,
  offset = TRUE,
  max_it = 100,
  link = "log",
  theta_estimate = "none",
  test = "chisq",
  BPPARAM = SerialParam()
)
```

Arguments

scmpObj	A ScMaSigPro object.
p_value	The level of the adjusted p-values: the genes at or below it have a non-flat profile. It is also the default level of sc.t.fit() .
mt_correction	The method of stats::p.adjust() that adjusts the p-values for multiple testing; "BH" is that of Benjamini and Hochberg (1995).
min_na	The minimum number of bins with a value (not NA) that a gene needs to be fitted, as min.obs of p.vector() in maSigPro. Only counts with missing values, such as microarray data given to create_scmp() with use_as_bin = TRUE, have fewer; it can be at most the number of bins of the design.
min_counts	Minimum total count of a gene, summed over the bins of the design. Genes with fewer counts are left out of the fit. The default, 0, keeps every gene.
min_bins_expressed	Minimum number of bins of the design in which a gene has a count above 0. Genes expressed in fewer bins are left out of the fit. The default, 0, keeps every gene; it can be at most the number of bins.

family	The distribution of the error term, a <code>stats::family()</code> object: by default the negative binomial with $\theta = 10$ and the link link. A family given here keeps its own link.
epsilon	The convergence tolerance of <code>stats::glm()</code> .
verbose	Whether to print progress messages.
offset	Whether to add the offset of each bin, the logarithm of its size factor (see Details), to the linear predictor; it needs the log link.
max_it	The maximum number of iterations of <code>stats::glm()</code> .
link	The link of the default family: "log", the model of the paper, or "identity". When family is given, link must be its link. The identity link needs <code>offset = FALSE</code> ; under it the fitted mean of a gene can become negative, and <code>stats::glm()</code> then warns.
theta_estimate	For a negative binomial family: "none" (the default) fits every gene with the theta of family (10 by default, as maSigPro). "gene" estimates theta by maximum likelihood on each gene's full model and fits the gene with it; "global" fits every gene with the median of these estimates. The estimates are kept in the distribution parameter, for <code>sc.t.fit()</code> and <code>plotDiagnostics()</code> : <code>theta_gene</code> holds one per gene and <code>theta_estimate</code> the choice.
test	The test of each gene's full model against its intercept-only model: "chisq" (the default) or "F". Both divide the difference in deviance between the two models by the dispersion of the full model, which, for the negative binomial and Gaussian families, is estimated as its Pearson statistic divided by its residual degrees of freedom. "chisq" compares the result with the chi-squared distribution, as maSigPro does. "F" also divides it by the difference in degrees of freedom and compares it with the F distribution on the residual degrees of freedom of the full model: the F test of <code>stats::anova.glm()</code> . With the Gaussian family both values give the F test, the test of maSigPro for that family. "F" needs a family whose dispersion is estimated, so not <code>stats::binomial()</code> or <code>stats::poisson()</code> . The value is kept in the test parameter. See Details.
BPPARAM	A BiocParallelParam object that runs the per-gene fits, such as <code>BiocParallel::MulticoreParam(2)</code> (Linux and macOS) or <code>BiocParallel::SnowParam(2)</code> (any system). The default, <code>BiocParallel::SerialParam()</code> , fits the genes one after another; create it with <code>progressbar = TRUE</code> for a progress bar. The results do not depend on the backend.

Details

With `offset = TRUE`, the fits include an offset as Srivastava et al. (2024), Section 2.3 and Supplementary Section S1.3, describe it: "scMaSigPro incorporates an offset value to the Poly-GLM model to adjust for disparities in total read counts per bin. This offset is calculated as the logarithm of size factors for each bin". The size factors are those of the median-of-ratios method (Anders and Huber 2010; Love et al. 2014) on the raw counts: "The size factor for each bin is the median of the ratios of its counts to the geometric means for each gene". Only the genes with a count above 0 in every bin enter the medians, and the fit stops when there is none: fewer, larger bins (`sc.squeeze()`) or `offset = FALSE` avoid it. The genes are fitted on their raw counts, without a pseudo-count: "scMaSigPro can directly process raw counts". The offsets are stored in the Design slot, where `sc.t.fit()` finds them.

Genes are left out of the fit when they have fewer than `min_na` values, when all their counts are zero, and when they are below `min_counts` or `min_bins_expressed`. They have no p-value and do not count in the multiple-testing correction; `scmpResults()` gives them NA. The size factors are computed before `min_counts` and `min_bins_expressed` are applied, so these two filters change the results of the genes they keep only through the correction.

The test of the full model is the first step of the method of Srivastava et al. (2024), Section 2.4: "scMaSigPro implements a two-step strategy similar to the maSigPro method", in which the full model of each gene is "evaluated against a simpler intercept-only model via hypothesis testing based on the log-likelihood ratio statistic". The default, `test = "chisq"`, is that test as maSigPro computes it for families other than the Gaussian: the statistic is divided by the estimated dispersion and compared with the chi-squared distribution, as if the dispersion were known. With few bins, and so few residual degrees of freedom, the estimate varies from gene to gene and the test calls more genes significant than its level. `test = "F"`, the quasi-likelihood F test, accounts for the estimation of the dispersion through the residual degrees of freedom of the F distribution.

Value

The object with the p-values and the genes with a non-flat profile in the Profile slot (class [VariableProfiles](#)), the offsets in the Design slot and the settings of the fit in the Parameters slot.

Author(s)

Priyansh Srivastava <spriyansh29@gmail.com>, Ana Conesa and Maria Jose Nueda, <mj.nueda@ua.es>

References

- Srivastava P, Benegas Coll M, Götz S, Nueda MJ, Conesa A (2024). scMaSigPro: differential expression analysis along single-cell trajectories. *Bioinformatics* 40(7), btac443. doi:10.1093/bioinformatics/btac443
- Conesa A, Nueda MJ, Ferrer A, Talón M (2006). maSigPro: a method to identify significantly differential expression profiles in time-course microarray experiments. *Bioinformatics* 22(9), 1096-1102. doi:10.1093/bioinformatics/btl056
- Nueda MJ, Tarazona S, Conesa A (2014). Next maSigPro: updating maSigPro bioconductor package for RNA-seq time series. *Bioinformatics* 30(18), 2598-2602. doi:10.1093/bioinformatics/btu333
- Anders S, Huber W (2010). Differential expression analysis for sequence count data. *Genome Biology* 11(10), R106. doi:10.1186/gb20101110r106
- Love MI, Huber W, Anders S (2014). Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. *Genome Biology* 15(12), 550. doi:10.1186/s1305901405508
- Benjamini Y, Hochberg Y (1995). Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society: Series B* 57(1), 289-300. doi:10.1111/j.25176161.1995.tb02031.x
- Venables WN, Ripley BD (2002). *Modern Applied Statistics with S*, fourth edition. Springer, New York. ISBN 0-387-95457-0.

See Also

[scmpResults\(\)](#) for the p-values; [VariableProfiles](#).

Other workflow steps: [sc.cluster.trend\(\)](#), [sc.filter\(\)](#), [sc.restruct\(\)](#), [sc.set.poly\(\)](#), [sc.squeeze\(\)](#), [sc.t.fit\(\)](#)

Examples

```
data("scmp.ob", package = "scMaSigPro")
# Test the full model of every gene with the F test
scmp <- sc.p.vector(scmp.ob, test = "F", verbose = FALSE)
sum(scmpResults(scmp)$non_flat)
```

sc.restruct

Restructure the bins of paths that share a root

Description

`sc.restruct()` rebuilds the bins so that every end path starts at the root: the bins of each link path are added to the end paths it links, the bins of the root to every end path, the binned pseudotime of the later bins is shifted to follow them, and [pb_counts\(\)](#) recomputes the pseudo-bulk counts.

Usage

```
sc.restruct(
  scmpObj,
  end_node_list,
  root_node,
  link_node_list,
  assay_name = "counts",
  aggregate = "sum",
  link_sep = "_links_",
  verbose = TRUE
)
```

Arguments

<code>scmpObj</code>	A ScMaSigPro object.
<code>end_node_list</code>	A list of the labels of the end paths.
<code>root_node</code>	The label of the root path.
<code>link_node_list</code>	A list of the labels of the link paths, the segments that some end paths share; a label holds the labels of the paths it links, separated by <code>link_sep</code> .
<code>assay_name</code>	The name of the assay of the Sparse slot that holds the counts of the cells.
<code>aggregate</code>	How the counts of the cells of a bin are combined: "sum" or "mean" (rounded to whole numbers).

link_sep	The separator in the labels of the link paths, matched as fixed text when the label contains it and as a regular expression otherwise, so " " and "\\ " both work.
verbose	Whether to print progress messages.

Value

The object with the bins of every end path, from the root on, and their pseudo-bulk counts in the Dense slot.

Author(s)

Priyansh Srivastava <spriyansh29@gmail.com>

See Also

[assign_paths\(\)](#), which labels the root and link paths.

Other workflow steps: [sc.cluster.trend\(\)](#), [sc.filter\(\)](#), [sc.p.vector\(\)](#), [sc.set.poly\(\)](#), [sc.squeeze\(\)](#), [sc.t.fit\(\)](#)

Examples

```
data("multi.lin.sce", package = "scMaSigPro")
scmp <- as_scmp(multi.lin.sce,
  additional_params = list(
    labels_exist = TRUE, exist_ptime_col = "Monocle3_Pseudotime",
    exist_path_col = "group"
  ),
  verbose = FALSE
)
scmp <- sc.squeeze(scmp)
# Three end paths, one link path shared by two of them, and the root
scmp <- sc.restruct(scmp,
  end_node_list = list("path_Y_15", "path_Y_18", "path_Y_52"),
  root_node = "root", link_node_list = list("path_Y_18|path_Y_15"),
  link_sep = "|", verbose = FALSE
)
table(cDense(scmp)$Path)
```

sc.set.poly

Set up the polynomial design

Description

sc.set.poly() builds the design of the maSigPro model (Conesa et al. 2006): a 0/1 dummy for every path but the first, the reference, and the products of each dummy with each power of the binned pseudotime up to poly_degree.

The reference is reference when it is given. Otherwise it is the first path of the Dense slot: after [sc.squeeze\(\)](#), the first level of a factor path column, or else the first label in C-locale (byte) order; after [sc.restruct\(\)](#), the first path of `end_node_list`; with bins given to [create_scmp\(\)](#) or [cDense\(\)](#), the path of the first bin. The other paths keep the order of the Dense slot.

Usage

```
sc.set.poly(
  scmpObj,
  poly_degree = 2,
  bin_ptime_col = scmpObj@Parameters@bin_ptime_col,
  path_col = scmpObj@Parameters@path_col,
  reference = NULL
)
```

Arguments

<code>scmpObj</code>	A ScMaSigPro object.
<code>poly_degree</code>	The degree of the polynomial in pseudotime.
<code>bin_ptime_col</code>	The name of the column of the bin metadata that holds the binned pseudotime of each bin.
<code>path_col</code>	The name of the column of the cell or bin metadata that holds the path label of each cell or bin.
<code>reference</code>	The reference path, one of the path labels of the Dense slot, or NULL for the first path.

Value

The object with the design in the Design slot (class [MatrixDesign](#)).

Author(s)

Priyansh Srivastava <spriyansh29@gmail.com>, Ana Conesa and Maria Jose Nueda, <mj.nueda@ua.es>

References

- Conesa A, Nueda MJ, Ferrer A, Talón M (2006). `maSigPro`: a method to identify significantly differential expression profiles in time-course microarray experiments. *Bioinformatics* 22(9), 1096-1102. doi:10.1093/bioinformatics/btl056
- Srivastava P, Benegas Coll M, Götz S, Nueda MJ, Conesa A (2024). `scMaSigPro`: differential expression analysis along single-cell trajectories. *Bioinformatics* 40(7), btae443. doi:10.1093/bioinformatics/btae443

See Also

[showPoly\(\)](#), [predictors\(\)](#) and [pathAssign\(\)](#), which read the design; [MatrixDesign](#).

Other workflow steps: [sc.cluster.trend\(\)](#), [sc.filter\(\)](#), [sc.p.vector\(\)](#), [sc.restruct\(\)](#), [sc.squeeze\(\)](#), [sc.t.fit\(\)](#)

Examples

```
data("scmp.ob", package = "scMaSigPro")
scmp <- sc.set.poly(scmp.ob, poly_degree = 2)
showPoly(scmp)
```

sc.squeeze

Bin the cells of each path along pseudotime

Description

`sc.squeeze()` divides the pseudotime range of each path into bins of equal width and adds up the counts of the cells of each bin into pseudo-bulk counts. The number of bins of a path is that of a histogram rule (`estBinSize()`) times `drop_fac`, rounded up.

Each cell goes to the bin whose interval holds its pseudotime. As in Srivastava et al. (2024), Section 2.1, "bins are structured as left-closed and right-open intervals, except for the last bin, which is closed on both ends": a cell on the bound of two bins goes to the later one. The bounds are stored in the `scmp_l_bound` and `scmp_u_bound` columns of the bin metadata (`cDense()`).

The paths are binned, and their bins stored, in the order of the levels of the path column when it is a factor, and otherwise in the C-locale (byte) order of the labels, whatever the collation of the session. `sc.set.poly()` takes the first path as the reference unless told otherwise.

Usage

```
sc.squeeze(
  scmpObj,
  ptime_col = scmpObj@Parameters@ptime_col,
  path_col = scmpObj@Parameters@path_col,
  bin_method = "Sturges",
  drop_fac = 1,
  verbose = FALSE,
  bin_mem_col = "scmp_bin_members",
  bin_col = "scmp_bin",
  bin_size_col = "scmp_bin_size",
  bin_ptime_col = "scmp_binned_pseudotime",
  split_bins = FALSE,
  prune_bins = FALSE,
  assay_name = "counts",
  drop_trails = FALSE,
  aggregate = "sum",
  fill_gaps = FALSE,
  additional_params = list(use_unique_time_points = FALSE)
)
```

Arguments

scmpObj	A ScMaSigPro object.
ptime_col	The name of the column of the cell metadata that holds the pseudotime of each cell.
path_col	The name of the column of the cell or bin metadata that holds the path label of each cell or bin.
bin_method	The rule for the number of bins: "Sturges", "Sqrt", "Rice", "Doane", "Freedman.Diaconis" or "Scott.Normal" (see estBinSize()).
drop_fac	The drop factor of Srivastava et al. (2024), Section 2.1, at least 0.3: it multiplies the number of bins of the rule, and the product is rounded up. Values below 1 give fewer, wider bins.
verbose	Whether to print progress messages.
bin_mem_col	The name of the column of the bin metadata that holds the cells of each bin, separated by .
bin_col	The name of the column of the bin metadata that holds the name of each bin.
bin_size_col	The name of the column of the bin metadata that holds the number of cells of each bin.
bin_ptime_col	The name of the column of the bin metadata that holds the binned pseudotime of each bin.
split_bins	Whether to divide the bins of a path that hold more cells than the mean plus the standard deviation of its bin sizes (see Details).
prune_bins	Whether to remove the bins of a path that hold fewer cells than the mean minus the standard deviation of its bin sizes (see Details).
assay_name	The name of the assay of the Sparse slot that holds the counts of the cells.
drop_trails	Whether to drop the last bins of the longer paths, so that every path has the binned pseudotime of the shortest.
aggregate	How the counts of the cells of a bin are combined: "sum" or "mean" (rounded to whole numbers).
fill_gaps	Whether to renumber the bins of a path whose binned pseudotime has gaps, such as after pruning.
additional_params	A list: use_unique_time_points = TRUE bins the distinct pseudotime values of each path instead of its cells.

Details

split_bins and prune_bins follow Section 2.2 of Srivastava et al. (2024): "scMaSigPro calculates the mean (μ) and standard deviation (σ) of the sizes of all bins per path to establish maximum ($\mu + \sigma$) and minimum ($\mu - \sigma$) allowable bin sizes. Bins larger than the maximum size are divided, and those smaller than the minimum size can be optionally removed, ensuring uniform bin sizes. The process is repeated until all bins conform to the established size criteria." Both limits are set once, from the bins of the rule, empty bins included. A bin above the maximum is divided into the number of bins that the rule and drop_fac give for its cells, until no bin is larger; sc.squeeze() stops when a division gives back bins it has already made. A path with a single bin keeps it. Pruning comes after the divisions: a negative minimum removes no bin, and the cells of a removed bin stay in the Sparse slot without a bin.

Value

The object with the bins and their pseudo-bulk counts (`bulk.counts`) in the Dense slot and the binned pseudotime of each cell in the cell metadata.

Author(s)

Priyansh Srivastava <spriyansh29@gmail.com>

References

Srivastava P, Benegas Coll M, Götz S, Nueda MJ, Conesa A (2024). scMaSigPro: differential expression analysis along single-cell trajectories. *Bioinformatics* 40(7), btae443. doi:10.1093/bioinformatics/btae443

See Also

`estBinSize()` for the rules that set the number of bins; `plotBinTile()` to see the bins.

Other workflow steps: `sc.cluster.trend()`, `sc.filter()`, `sc.p.vector()`, `sc.restruct()`, `sc.set.poly()`, `sc.t.fit()`

Examples

```
data("splat.sim", package = "scMaSigPro")
scmp <- as_scmp(splat.sim,
  additional_params = list(
    labels_exist = TRUE, exist_ptime_col = "Step",
    exist_path_col = "Group"
  ),
  verbose = FALSE
)
scmp <- sc.squeeze(scmp)
bins <- cDense(scmp)
head(bins[, c("Path", "scmp_binned_pseudotime", "scmp_bin_size")])
```

sc.t.fit

Select the significant terms of each gene

Description

`sc.t.fit()` selects by stepwise regression, as maSigPro does, the significant terms of the full model of each gene with a non-flat profile (`sc.p.vector()`), and keeps the coefficients, p-values and t statistics of the selected models.

Usage

```

sc.t.fit(
  scmpObj,
  selection_method = "backward",
  p_value = scmpObj@Parameters@p_value,
  nvar_correction = FALSE,
  family = scmpObj@Parameters@distribution,
  epsilon = scmpObj@Parameters@epsilon,
  offset = scmpObj@Parameters@offset,
  verbose = TRUE,
  max_it = scmpObj@Parameters@max_it,
  link = scmpObj@Parameters@link,
  BPPARAM = SerialParam()
)

```

Arguments

scmpObj	A ScMaSigPro object.
selection_method	The stepwise method: "backward", "forward", "two.ways.backward" or "two.ways.forward".
p_value	The level at which a term enters or stays in a model; by default that of sc.p.vector() .
nvar_correction	Whether to divide p_value by the number of terms of the design.
family	The distribution of the error term, a stats::family() object; by default the family of sc.p.vector() , with its link. When sc.p.vector() estimated one theta per gene (theta_estimate = "gene"), each gene is fitted with its own theta.
epsilon	The convergence tolerance of stats::glm() .
offset	Whether to add the offset of each bin, the logarithm of its size factor, which needs the log link. The fits use the offsets that sc.p.vector() stored in the Design slot; when it stored none (offset = FALSE), they are computed here in the same way.
verbose	Whether to print progress messages.
max_it	The maximum number of iterations of stats::glm() .
link	The link of family, by default that of sc.p.vector() ; when given, it must be the link of family. Choose the link in sc.p.vector() .
BPPARAM	A BiocParallelParam object that runs the per-gene fits, such as BiocParallel::MulticoreParam(2) (Linux and macOS) or BiocParallel::SnowParam(2) (any system). The default, BiocParallel::SerialParam() , fits the genes one after another; create it with <code>progressbar = TRUE</code> for a progress bar. The results do not depend on the backend.

Value

The object with the selected model of each gene with a non-flat profile in the Estimate slot (class [Estimates](#)), read by [showSol\(\)](#) and the other result readers.

Author(s)

Priyansh Srivastava <spriyansh29@gmail.com>, Ana Conesa and Maria Jose Nueda, <mj.nueda@ua.es>

References

Conesa A, Nueda MJ, Ferrer A, Talón M (2006). maSigPro: a method to identify significantly differential expression profiles in time-course microarray experiments. *Bioinformatics* 22(9), 1096-1102. doi:10.1093/bioinformatics/btl056

Nueda MJ, Tarazona S, Conesa A (2014). Next maSigPro: updating maSigPro bioconductor package for RNA-seq time series. *Bioinformatics* 30(18), 2598-2602. doi:10.1093/bioinformatics/btu333

Srivastava P, Benegas Coll M, Götz S, Nueda MJ, Conesa A (2024). scMaSigPro: differential expression analysis along single-cell trajectories. *Bioinformatics* 40(7), btae443. doi:10.1093/bioinformatics/btae443

See Also

[showSol\(\)](#) and the other readers of the results; [Estimates](#).

Other workflow steps: [sc.cluster.trend\(\)](#), [sc.filter\(\)](#), [sc.p.vector\(\)](#), [sc.restruct\(\)](#), [sc.set.poly\(\)](#), [sc.squeeze\(\)](#)

Examples

```
data("scmp.ob", package = "scMaSigPro")
# Keep the terms with a p-value below 0.01
scmp <- sc.t.fit(scmp.ob, p_value = 0.01, verbose = FALSE)
head(showSol(scmp)[, c("p-value", "R-squared")])
```

ScMaSigPro

ScMaSigPro

Description

An analysis of the scMaSigPro workflow: the data at the level of the cells and of the pseudotime bins, as two SingleCellExperiment objects, and the results and settings of every step.

Details

[validObject\(\)](#) checks every slot with the validity method of its class and reports each problem with the name of the slot; version must be a single version number. The constructors, the steps of the workflow, [pb_counts\(\)](#) and the setters check the object they return, so a step or setter that would return an invalid object stops with these problems instead. Objects saved by an earlier version of the package, for example with [saveRDS\(\)](#), are brought up to date with [updateObject\(\)](#).

Value

An object of class `ScMaSigPro`. `as_scmp()` and `create_scmp()` make one with the cells in the Sparse slot; `sc.squeeze()` adds the bins (Dense), `sc.set.poly()` the design (Design), `sc.p.vector()` the test of each gene (Profile), `sc.t.fit()` the selected models (Estimate), `sc.filter()` and `sc.cluster.trend()` the gene lists and clusters (Significant), and every step its settings (Parameters).

Slots

Sparse The cells: a `SingleCellExperiment` with their counts and metadata, including the pseudotime and path columns.

Dense The pseudotime bins: a `SingleCellExperiment` with their pseudo-bulk counts and metadata.

Design The design, a `MatrixDesign` object.

Profile The test of each gene, a `VariableProfiles` object.

Estimate The selected models, an `Estimates` object.

Significant The gene lists and clusters, a `Significant` object.

Parameters The settings, a `ParameterConfig` object.

version The version of `scMaSigPro` that created the object or last changed its structure in `updateObject()`.

Why a class of its own

`ScMaSigPro` holds two `SingleCellExperiment` objects because the analysis works at two resolutions whose columns differ. **Sparse** holds the cells, with their pseudotime and path labels. **Dense** holds the pseudotime bins (the pseudo-bulk) to which the polynomial GLMs are fitted. One `SingleCellExperiment` cannot hold both: `altExp()` needs the columns of the main experiment, and bins stored in `metadata()` would lose the validity checks and accessors of an experiment. The other slots hold results that have no place in the `SummarizedExperiment` model: the design matrices (`MatrixDesign`), the per-gene fits of the full model (`VariableProfiles`), the coefficients and significance of the selected models (`Estimates`), the gene lists and clusters (`Significant`) and a record of every parameter used (`ParameterConfig`), for reproducibility. Both experiments are built and changed through the `SummarizedExperiment` accessors only, and `validObject()` runs after every step of the workflow. To use the results with other tools, `scmpResults()` returns them as a `DataFrame` with one row per gene, and `scmpToRowData()` writes them into the `rowData()` of a `SingleCellExperiment`.

See Also

Other classes: `Estimates`, `MatrixDesign`, `ParameterConfig`, `Significant`, `VariableProfiles`, `updateObject`, `ScMaSigPro`-method

Examples

```
data("scmp.ob", package = "scMaSigPro")
scmp.ob
validObject(scmp.ob)
```

Description

A [ScMaSigPro](#) object with every step of the workflow run on [splat.sim](#), for the examples and the vignettes. The **Quick start** vignette runs the same steps.

Usage

```
data("scmp.ob", package = "scMaSigPro")
```

Format

A [ScMaSigPro](#) object:

Sparse The 200 cells and 100 genes of [splat.sim](#), with the pseudotime in Pseudotime (Step of [splat.sim](#)) and the path in Path (Group).

Dense The 16 bins of [sc.squeeze\(\)](#) with its defaults, 8 per path, and their pseudo-bulk counts.

Design The design of [sc.set.poly\(\)](#) with `poly_degree = 1` and the offsets of [sc.p.vector\(\)](#).

Profile The test of the full model of every gene by [sc.p.vector\(\)](#): negative binomial family with theta 10, log link and size-factor offsets.

Estimate The models selected by [sc.t.fit\(\)](#) (backward selection).

Significant The gene lists Path1 and Path2vsPath1 of [sc.filter\(\)](#) with `rsq = 0.7` and `vars = "groups"`, and four clusters of their union from [sc.cluster.trend\(\)](#).

Parameters The settings of these steps.

Author(s)

Priyansh Srivastava <spriyansh29@gmail.com>

Source

Made by `system.file("scripts", "make-scmp-ob.R", package = "scMaSigPro")` from [splat.sim](#).

References

Srivastava P, Benegas Coll M, Götz S, Nueda MJ, Conesa A (2024). scMaSigPro: differential expression analysis along single-cell trajectories. *Bioinformatics* 40(7), btae443. doi:10.1093/bioinformatics/btae443

See Also

Other datasets: [multi.lin.sce](#), [splat.sim](#)

Examples

```
data("scmp.ob", package = "scMaSigPro")
scmp.ob
```

scmpResults	<i>Per-gene results of an analysis</i>
-------------	--

Description

scmpResults() collects the per-gene results stored in a [ScMaSigPro](#) object in one table with a row per gene. scmpToRowData() writes that table into the rowData() of a SingleCellExperiment, such as the one the analysis started from, so that the results travel with the data.

Usage

```
scmpResults(scmpObj)

scmpToRowData(sce, scmpObj)
```

Arguments

scmpObj	A ScMaSigPro object.
sce	A SingleCellExperiment, or another SummarizedExperiment, whose row names are gene names.

Details

scmpResults() has one row per gene of the object (the rows of the Sparse slot), named by gene, and the columns of the steps that have been run:

- [sc.p.vector\(\)](#): p_value and adj_p_value, the p-value of the full model and its adjusted value, NA for the genes that were not fitted (too few values or only zero counts); non_flat, TRUE for the genes it selected.
- [sc.t.fit\(\)](#): model_p_value and r_squared, the p-value and the R-squared of the model selected by the stepwise regression (the p-value and R-squared columns of [showSol\(\)](#)), NA for the genes without a row there; influential, TRUE for the genes with influential observations (the columns of [showInflu\(\)](#)).
- [sc.filter\(\)](#): one column per gene list, named sig_ followed by the name of the list, TRUE for the genes of the list.
- [sc.cluster.trend\(\)](#): cluster, the cluster of each clustered gene, NA for the other genes.

The logical columns are FALSE, never NA, for the genes outside a set.

scmpToRowData() matches the genes by name. It removes the columns of rowData(sce) whose names start with scmp_, as an earlier call writes them, and adds the columns of scmpResults() with that prefix, such as scmp_p_value. The genes of sce that are not in the object get NA in every new column, and the genes of the object that are not in sce are left out. It stops if no gene matches.

Value

scmpResults(): a [DataFrame](#). scmpToRowData(): sce with the new rowData() columns.

Author(s)

Priyansh Srivastava <spriyansh29@gmail.com>

See Also

Other results: [queryCoeff\(\)](#), [showCoeff\(\)](#), [showPoly\(\)](#)

Examples

```
data("scmp.ob", package = "scMaSigPro")
res <- scmpResults(scmp.ob)
head(res)
# The genes of the "Path1" list, ordered by R-squared
sig <- res[res$sig_Path1, ]
head(sig[order(sig$r_squared, decreasing = TRUE), ])

# scmp.ob was made from the bundled splat.sim data set
data("splat.sim", package = "scMaSigPro")
sce <- scmpToRowData(splat.sim, scmp.ob)
head(SummarizedExperiment::rowData(sce)[, c("Gene", "scmp_p_value")])
```

showCoeff

Results of the term selection

Description

Read the per-gene results that [sc.t.fit\(\)](#) stores in a [ScMaSigPro](#) object:

- [showSol\(\)](#): for each gene whose selected model has a significant coefficient, the p-value and R-squared of that model and the p-value of each coefficient of the full model (NA for the terms it leaves out).
- [showCoeff\(\)](#): the coefficients of these genes, 0 for the terms the model leaves out and NA for the terms that cannot be estimated.
- [showTS\(\)](#): their t statistics.
- [showGroupCoeff\(\)](#): the coefficients of the polynomial of each path.
- [showInflu\(\)](#): the genes with influential observations, one column per gene and one row per bin; an entry names the influence measures that flag the bin (DFFITS and Cook's distance, see [stats::influence.measures\(\)](#)).
- [showSigProf\(\)](#): the pseudo-bulk counts of the genes of [showSol\(\)](#).

Usage

```

showCoeff(scmpObj, includeInflu = TRUE)

showInflu(scmpObj)

showTS(scmpObj, includeInflu = TRUE)

showSol(scmpObj, includeInflu = TRUE)

showSigProf(scmpObj, includeInflu = FALSE)

showGroupCoeff(scmpObj, includeInflu = TRUE)

```

Arguments

scmpObj A [ScMaSigPro](#) object.

includeInflu Whether to keep the genes with influential observations, the columns of `showInflu()`.

Value

`showSol()`, `showCoeff()` and `showGroupCoeff()`: a data frame with one row per gene. `showTS()`: a numeric matrix with one row per gene. `showInflu()`: a character matrix with one row per bin and one column per gene, 0 x 0 when no gene has influential observations. `showSigProf()`: a `dgCMatrix` with one row per gene and one column per bin.

See Also

Other results: [queryCoeff\(\)](#), [scmpResults\(\)](#), [showPoly\(\)](#)

Examples

```

data("scmp.ob", package = "scMaSigPro")
head(showSol(scmp.ob))
head(showGroupCoeff(scmp.ob))
colnames(showInflu(scmp.ob))

```

showPoly

Parameters and model formula of an analysis

Description

`showParams()` lists the parameters that the steps of the workflow store in a [ScMaSigPro](#) object (see [ParameterConfig](#)). `showPoly()` writes the full model that [sc.set.poly\(\)](#) set up, with one coefficient `beta1`, `beta2`, ... per column of the design matrix.

Usage

```
showPoly(scmpObj)

showParams(scmpObj)
```

Arguments

scmpObj A [ScMaSigPro](#) object.

Value

showParams(): a data frame with the columns parameters and value, one row per parameter.
showPoly(): a character string.

See Also

Other results: [queryCoeff\(\)](#), [scmpResults\(\)](#), [showCoeff\(\)](#)

Examples

```
data("scmp.ob", package = "scMaSigPro")
head(showParams(scmp.ob))
showPoly(scmp.ob)
```

Significant	<i>Significant</i>
-------------	--------------------

Description

The gene lists of [sc.filter\(\)](#) and the clusters of [sc.cluster.trend\(\)](#).

Details

validObject() checks that every element of genes is a character vector of gene names and that clusters holds one value per gene, named by gene.

Value

An object of class Significant: the gene lists of [sc.filter\(\)](#) and the clusters of [sc.cluster.trend\(\)](#), stored in the Significant slot of a [ScMaSigPro](#) object. [scmpResults\(\)](#) returns them per gene.

Slots

genes A named list of gene lists.
clusters A list with the cluster of each clustered gene, named by gene.

See Also

Other classes: [Estimates](#), [MatrixDesign](#), [ParameterConfig](#), [ScMaSigPro](#), [VariableProfiles](#), [updateObject,ScMaSigPro-method](#)

Examples

```
data("scmp.ob", package = "scMaSigPro")
table(scmpResults(scmp.ob)$cluster)
```

splat.sim	<i>Simulated bifurcating trajectory</i>
-----------	---

Description

A SingleCellExperiment with the counts of 100 genes in 200 cells, simulated with splatter (Zappia et al. 2017) along two paths that start from the same point. The vignettes [Quick start](#) and [Model and tuning](#) analyse it.

Usage

```
data("splat.sim", package = "scMaSigPro")
```

Format

A SingleCellExperiment with 100 genes (rows) and 200 cells (columns):

counts **assay** The simulated counts.

colData Cell, the cell name; Batch, the batch (one, "Batch1"); Group, the path of the cell, a factor with the levels "Path1" and "Path2"; ExpLibSize, its expected library size; Step, how far along its path the cell is (1 to 100), used as the pseudotime; sizeFactor, the size factor of scuttle::logNormCounts().

rowData Gene, the gene name; BaseGeneMean, the base mean of the gene; OutlierFactor, its outlier factor; GeneMean, its mean after the outlier factor; LocalDEFacPath1 and LocalDEFacPath2, its differential expression factor along each path; DEFacPath1 and DEFacPath2, its factor at the end of each path; SigmaFacPath1 and SigmaFacPath2, the factor of its non-linear change along each path (0 for a linear change).

reducedDims PCA, the first two principal components of the log-normalised counts.

Author(s)

Priyansh Srivastava <spriyansh29@gmail.com>

Source

Simulated with splatter::splatSimulatePaths() (splatter 1.26.0, set.seed(123), 200 cells, 100 genes, two paths of 100 steps from the origin, de.prob = 0.3, de.facLoc = 0.2), normalised with scuttle::logNormCounts() and reduced with scater::runPCA(). The script is system.file("scripts", "make-splat-sim.R", package = "scMaSigPro").

References

Zappia L, Phipson B, Oshlack A (2017). Splatter: simulation of single-cell RNA sequencing data. *Genome Biology* 18, 174. doi:10.1186/s1305901713050

See Also

Other datasets: [multi.lin.sce](#), [scmp.ob](#)

Examples

```
data("splat.sim", package = "scMaSigPro")
table(SummarizedExperiment::colData(splat.sim)$Group)
```

updateObject, ScMaSigPro-method

Update an ScMaSigPro object to the current class definition

Description

updateObject() brings a [ScMaSigPro](#) object saved by an earlier version of scMaSigPro, for example with saveRDS() or save(), up to date. Call it on such an object after readRDS() or load(), before the next step of the workflow.

Usage

```
## S4 method for signature 'ScMaSigPro'
updateObject(object, ..., verbose = FALSE)
```

Arguments

object	A ScMaSigPro object.
...	Passed to the updateObject() methods of the Sparse and Dense slots.
verbose	Whether to print a message when the object is updated.

Details

Objects saved before the version slot was added (scMaSigPro 0.0.5 and earlier) fail validObject(); updateObject() adds the slot with the version of the package that is running. For every object, the Sparse and Dense slots are passed to the updateObject() method of SingleCellExperiment, and the result is checked with validObject(). A current object, such as one created by this version of the package, is returned unchanged.

Objects saved before the min_counts and min_bins_expressed parameters existed get both set to 0, the values that keep every gene ([sc.p.vector\(\)](#) had no such filter then), and the version of the package that is running.

Objects saved before the test parameter existed get it set to "chisq", the test of `sc.p.vector()` then, and the version of the package that is running.

Objects saved with the `log_offset` parameter lose it. The fits of an object saved with `log_offset = FALSE`, the default, used the size factor of each bin as the offset: `updateObject()` stores the logarithm of the size factors instead, the offset of `sc.p.vector()` and `sc.t.fit()`, and warns that the fits in the Profile and Estimate slots were made with the size factors, until `sc.p.vector()` is run again. The offsets of an object saved with `log_offset = TRUE` are kept. An object whose fits used no offset, or that was not fitted, keeps no offset: the offsets that an earlier fit left in it, or the zeros of `sc.set.poly()`, are removed.

Objects fitted with `link = "identity"` by `scMaSigPro` 0.0.5 or earlier hold a family labelled with the identity link but with the link functions of the log link, with which their fits were made: `updateObject()` warns, and `sc.t.fit()` stops on them, until `sc.p.vector()` is run again.

Objects binned by `scMaSigPro` 0.0.4 cannot be updated: their Dense slot is not a valid `SingleCellExperiment`, and the cell metadata of their Sparse slot may not be in the order of the counts. `updateObject()` stops with the command that bins the cells of the Sparse slot again with `sc.squeeze()`; the later steps of the workflow must then be run again.

Value

The updated `ScMaSigPro` object.

Author(s)

Priyansh Srivastava <spriyansh29@gmail.com>

See Also

[ScMaSigPro](#)

Other classes: [Estimates](#), [MatrixDesign](#), [ParameterConfig](#), [ScMaSigPro](#), [Significant](#), [VariableProfiles](#)

Examples

```
data("scmp.ob", package = "scMaSigPro")
path <- tempfile(fileext = ".rds")
saveRDS(scmp.ob, path)

# Later, possibly with a newer version of scMaSigPro
ob <- updateObject(readRDS(path))
unlink(path)
```

VariableProfiles	<i>VariableProfiles</i>
------------------	-------------------------

Description

The test of the full model of each gene by `sc.p.vector()`.

Details

`validObject()` checks that `p_values` and `adj_p_values` have the same length and names, that `fdr` holds at most one value and that every gene in `non_flat` is named in `p_values`.

Value

An object of class `VariableProfiles`: the test of the full model of each gene, which `sc.p.vector()` stores in the `Profile` slot of a `ScMaSigPro` object. `scmpResults()` returns its p-values.

Slots

`non_flat` The genes with a non-flat profile.
`p_values` The p-value of every fitted gene, named by gene.
`adj_p_values` The adjusted p-values, in the same order.
`fdr` The largest p-value of the genes with a non-flat profile, empty when there is none.

Author(s)

Priyansh Srivastava spriyansh29@gmail.com

See Also

Other classes: [Estimates](#), [MatrixDesign](#), [ParameterConfig](#), [ScMaSigPro](#), [Significant](#), [updateObject](#), [ScMaSigPro-me](#)

Examples

```
data("scmp.ob", package = "scMaSigPro")
res <- scmpResults(scmp.ob)
head(res[, c("p_value", "adj_p_value", "non_flat")])
```

Index

- * **accessors**
 - cDense, [6](#)
 - cSparse, [9](#)
 - eDense, [10](#)
 - eSparse, [11](#)
 - pathAssign, [17](#)
 - predictors, [27](#)
 - * **binning helpers**
 - pb_counts, [20](#)
 - * **classes**
 - Estimates, [13](#)
 - MatrixDesign, [14](#)
 - ParameterConfig, [16](#)
 - ScMaSigPro, [43](#)
 - Significant, [49](#)
 - updateObject, ScMaSigPro-method, [51](#)
 - VariableProfiles, [53](#)
 - * **datasets**
 - multi.lin.sce, [15](#)
 - scmp.ob, [45](#)
 - splat.sim, [50](#)
 - * **input helpers**
 - as_scmp, [3](#)
 - assign_paths, [4](#)
 - create_scmp, [8](#)
 - paths_from_graph, [18](#)
 - * **models**
 - sc.p.vector, [33](#)
 - sc.t.fit, [41](#)
 - * **plots**
 - plotBinTile, [21](#)
 - plotDiagnostics, [22](#)
 - plotIntersect, [23](#)
 - plotTrend, [24](#)
 - plotTrendCluster, [26](#)
 - * **regression**
 - sc.p.vector, [33](#)
 - sc.t.fit, [41](#)
 - * **results**
 - queryCoeff, [28](#)
 - scmpResults, [46](#)
 - showCoeff, [47](#)
 - showPoly, [48](#)
 - * **workflow steps**
 - sc.cluster.trend, [29](#)
 - sc.filter, [31](#)
 - sc.p.vector, [33](#)
 - sc.restruct, [36](#)
 - sc.set.poly, [37](#)
 - sc.squeeze, [39](#)
 - sc.t.fit, [41](#)
- as_scmp, [3](#)
as_scmp(), [5](#), [6](#), [9](#), [16](#), [19](#), [44](#)
assign_paths, [4](#)
assign_paths(), [3](#), [4](#), [8](#), [9](#), [18](#), [19](#), [37](#)
- calc_bin_size(), [21](#)
cDense, [6](#)
cDense(), [10–12](#), [18](#), [20](#), [21](#), [28](#), [38](#), [39](#)
cDense, ScMaSigPro-method (cDense), [6](#)
cDense<- (cDense), [6](#)
cDense<- , ScMaSigPro-method (cDense), [6](#)
create_scmp, [8](#)
create_scmp(), [3–6](#), [19](#), [33](#), [38](#), [44](#)
cSparse, [9](#)
cSparse(), [7](#), [11](#), [12](#), [18](#), [28](#)
cSparse, ScMaSigPro-method (cSparse), [9](#)
cSparse<- (cSparse), [9](#)
cSparse<- , ScMaSigPro-method (cSparse), [9](#)
- DataFrame, [47](#)
- eDense, [10](#)
eDense(), [7](#), [10](#), [12](#), [18](#), [28](#)
eDense, ScMaSigPro-method (eDense), [10](#)
eDense<- (eDense), [10](#)
eDense<- , ScMaSigPro, character-method (eDense), [10](#)

- eDense<- , ScMaSigPro, missing-method (eDense), 10
- eSparse, 11
- eSparse(), 7, 10, 11, 18, 28
- eSparse, ScMaSigPro-method (eSparse), 11
- eSparse<- (eSparse), 11
- eSparse<- , ScMaSigPro, character-method (eSparse), 11
- eSparse<- , ScMaSigPro, missing-method (eSparse), 11
- estBinSize(), 21, 39–41
- Estimates, 13, 14, 17, 42–44, 50, 52, 53
- Estimates-class (Estimates), 13
- extract_interval(), 21

- MatrixDesign, 13, 14, 17, 38, 44, 50, 52, 53
- MatrixDesign-class (MatrixDesign), 14
- multi.lin.sce, 15, 45, 51

- optimize_bin_max(), 21

- ParameterConfig, 13, 14, 16, 44, 48, 50, 52, 53
- ParameterConfig-class (ParameterConfig), 16
- pathAssign, 17
- pathAssign(), 7, 10–12, 14, 28, 38
- pathAssign, ScMaSigPro-method (pathAssign), 17
- pathAssign<- (pathAssign), 17
- pathAssign<- , ScMaSigPro-method (pathAssign), 17
- paths_from_graph, 18
- paths_from_graph(), 3–6, 9
- pb_counts, 20
- pb_counts(), 7, 9, 36, 43
- plotBinTile, 21
- plotBinTile(), 23–25, 27, 41
- plotDiagnostics, 22
- plotDiagnostics(), 22, 24, 25, 27, 34
- plotIntersect, 23
- plotIntersect(), 22, 23, 25, 27, 32
- plotTrend, 24
- plotTrend(), 22–24, 26, 27
- plotTrendCluster, 26
- plotTrendCluster(), 22–25, 31
- predictors, 27
- predictors(), 7, 10–12, 14, 18, 38

- predictors, ScMaSigPro-method (predictors), 27
- predictors<- (predictors), 27
- predictors<- , ScMaSigPro-method (predictors), 27

- queryCoeff, 28
- queryCoeff(), 47–49

- sc.cluster.trend, 29
- sc.cluster.trend(), 17, 26, 32, 36–38, 41, 43–46, 49
- sc.filter, 31
- sc.filter(), 22, 23, 25, 26, 29–31, 36–38, 41, 43–46, 49
- sc.p.vector, 33
- sc.p.vector(), 14, 16, 17, 29, 31, 32, 37, 38, 41–46, 51–53
- sc.restruct, 36
- sc.restruct(), 4–6, 9, 20, 31, 32, 36, 38, 41, 43
- sc.set.poly, 37
- sc.set.poly(), 9, 14, 17, 27, 31, 32, 36, 37, 39, 41, 43–45, 48, 52
- sc.squeeze, 39
- sc.squeeze(), 9, 10, 16, 20, 31, 32, 34, 36–38, 43–45, 52
- sc.t.fit, 41
- sc.t.fit(), 13, 16, 17, 22, 24, 28, 31–34, 36–38, 41, 44–47, 52
- ScMaSigPro, 3, 6–14, 16–18, 20, 22, 23, 25–30, 32, 33, 36, 38, 40, 42, 43, 45–53
- ScMaSigPro-class (ScMaSigPro), 43
- scmp.ob, 16, 45, 51
- scmpResults, 46
- scmpResults(), 29, 35, 36, 44, 48, 49, 53
- scmpToRowData (scmpResults), 46
- scmpToRowData(), 44
- showCoeff, 47
- showCoeff(), 13, 29, 47, 49
- showGroupCoeff (showCoeff), 47
- showGroupCoeff(), 13
- showInflu (showCoeff), 47
- showInflu(), 13, 30, 32, 46
- showParams (showPoly), 48
- showParams(), 16
- showPoly, 48
- showPoly(), 29, 38, 47, 48

showSigProf (showCoeff), 47
showSol (showCoeff), 47
showSol(), 13, 42, 43, 46
showTS (showCoeff), 47
showTS(), 13
Significant, 13, 14, 17, 30, 32, 44, 49, 52, 53
Significant-class (Significant), 49
splat.sim, 16, 45, 50
stats::anova.glm(), 34
stats::binomial(), 34
stats::dist(), 30
stats::family(), 16, 17, 34, 42
stats::glm(), 34, 42
stats::hclust(), 30
stats::influence.measures(), 47
stats::kmeans(), 30
stats::p.adjust(), 33
stats::poisson(), 34

updateObject(), 43, 44
updateObject, ScMaSigPro-method, 51

VariableProfiles, 13, 14, 17, 35, 36, 44, 50,
52, 53
VariableProfiles-class
(VariableProfiles), 53