Package 'SingleCellSignalR'

June 5, 2026

Title Cell Signalling Using Single-Cell RNA-seq or Proteomics Data

Version 2.3.0

Description Inference of ligand-receptor (L-R) interactions from single-cell expression (transcriptomics/proteomics) data.
SingleCellSignalR v2 inferences rely on the statistical model we introduced in the BulkSignalR package as well as the original SingleCellSignalR LR-score (both are available).
SingleCellSignalR v2 can be regarded as a wrapper to BulkSignalR fundamental classes. This also enables v2 users to work with any species, whereas only Mus musculus & Homo sapiens were available before in SingleCellSignalR v1.

URL https://github.com/jcolinge/SingleCellSignalR

BugReports https://github.com/jcolinge/SingleCellSignalR/issues

License CeCILL | file LICENSE

Depends R (>= 4.5)

Encoding UTF-8

LazyData FALSE

biocViews Network, RNASeq, Software, Proteomics, Transcriptomics, SingleCell, NetworkInference

Imports stats, utils, methods, ggplot2, matrixTests, matrixStats, foreach, BulkSignalR, grid, ComplexHeatmap, circlize

Suggests knitr, markdown, rmarkdown

RoxygenNote 7.3.3

VignetteBuilder knitr

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

git_branch devel

git_last_commit 8b72731

git_last_commit_date 2026-04-28

Repository Bioconductor 3.24

Date/Publication 2026-06-04

Author Jacques Colinge [aut] (ORCID: <https://orcid.org/0000-0003-2466-4824>), Jean-Philippe Villemin [cre] (ORCID: <https://orcid.org/0000-0002-1838-5880>)

Maintainer Jean-Philippe Villemin <jpvillemin@gmail.com>

Contents

`autocrines` *autocrines accessor*

Description

autocrines accessor

autocrines accessor

Usage

```
## S4 method for signature 'SCSRNet'
autocrines(x)

## S4 method for signature 'SCSRNoNet'
autocrines(x)
```

Arguments

x	SCSRNoNet object

Value

autocrines

autocrines

Examples

```
print("autocrines")
data(example_dataset, package = "SingleCellSignalR")
mat <- log1p(data.matrix(example_dataset[, -1])) / log(2)
rownames(mat) <- example_dataset[[1]]
rme <- rowMeans(mat)
mmat <- mat[rme > 0.05, ]
d <- dist(t(mmat))
h <- hclust(d, method = "ward.D")
pop <- paste0("pop_", cutree(h, 5))
scsrcn <- SCSRNet(mat,
    normalize = FALSE, method = "log-only",
    log.transformed = TRUE, populations = pop
)

autocrines(scsrcn)
```

---

autocrines<-,SCSRNet-method
autocrines setter (internal use only)

Description

autocrines setter (internal use only)

Usage

```
## S4 replacement method for signature 'SCSRNet'
autocrines(x) <- value
```

Arguments

x	SCSRNet object
value	value to be set for autocrines

Value

returns NULL

autocrines<-,SCSRNoNet-method *autocrines setter (internal use only)*

Description

autocrines setter (internal use only)

Usage

```
## S4 replacement method for signature 'SCSRNoNet'
autocrines(x) <- value
```

Arguments

x	SCSRNoNet object
value	value to be set for autocrines

Value

returns NULL

bsrdmComp *BSRDataModelComp object accessor*

Description

BSRDataModelComp object accessor

BSRDataModelComp object accessor

Usage

```
## S4 method for signature 'SCSRNet'
bsrdmComp(x)

## S4 method for signature 'SCSRNoNet'
bsrdmComp(x)
```

Arguments

x	SCSRNoNet object

Value

bsrdmComp

bsrdmComp object

Examples

```
data(example_dataset, package = "SingleCellSignalR")
mat <- log1p(data.matrix(example_dataset[, -1])) / log(2)
rownames(mat) <- example_dataset[[1]]
rme <- rowMeans(mat)
mmat <- mat[rme > 0.05, ]
d <- dist(t(mmat))
h <- hclust(d, method = "ward.D")
clusters <- paste0("pop_", cutree(h, 5))

scsrnn <- SCSRNet(mat,
    normalize = FALSE, method = "log-only",
    min.count = 1, prop = 0.001,
    log.transformed = TRUE, populations = clusters
)

bsrdmComp(scsrnn)

print("bsrdmComp")
data(example_dataset, package = "SingleCellSignalR")
mat <- log1p(data.matrix(example_dataset[, -1])) / log(2)
rownames(mat) <- example_dataset[[1]]
rme <- rowMeans(mat)
mmat <- mat[rme > 0.05, ]
d <- dist(t(mmat))
h <- hclust(d, method = "ward.D")
pop <- paste0("pop_", cutree(h, 5))
scsrnn <- SCSRNoNet(mat,
    normalize = FALSE, method = "log-only",
    log.transformed = TRUE, populations = pop
)
bsrdmComp(scsrnn)
```

---

bsrdmComp<-,SCSRNet-method

BSRDataModelComp object setter (internal use only)

Description

BSRDataModelComp object setter (internal use only)

Usage

```
## S4 replacement method for signature 'SCSRNet'
bsrdmComp(x) <- value
```

Arguments

x	SCSRNet object
value	value to be set for bsrdm.comp

Value

returns NULL

bsrdmComp<-,SCSRNoNet-method *BSRDataModelComp object setter (internal use only)*

Description

BSRDataModelComp object setter (internal use only)

Usage

```
## S4 replacement method for signature 'SCSRNoNet'
bsrdmComp(x) <- value
```

Arguments

x	SCSRNoNet object
value	value to be set for bsrdm.comp

Value

returns NULL

cellNetBubblePlot *Overview of cellular networks with a bubble plot*

Description

Overview of cellular networks with a bubble plot

Usage

```
cellNetBubblePlot(
  obj,
  selected.populations = NULL,
  genes.to.count = NULL,
  only.R.in.genes = TRUE,
  use.proportions = FALSE,
  low.color = "gray25",
  high.color = "firebrick1"
)
```

Arguments

obj	A SCSRNet or SCSRNoNet object.
selected.populations	A vector of cell population names to consider in the plot. By default, all the populations are considered.
genes.to.count	A vector of gene names for counting or enrichment analysis.

only.R.in.genes
: A logical indicating whether genes.to.count should be regarded as containing receptor gene names only.

use.proportions
: A logical to choose between representing the proportion of genes in genes.to.count or its enrichment.

low.color
: The color to be used when no gene list is provided or when the proportion of genes in genes.to.count is 0.

high.color
: The color for maximum proportion or best enrichment P-value.

Details

A matrix dot plot is generated to represent how much each pair of cell populations interact including autocrine interactions when available.

By default, the plot only reports the number of interactions, but it is possible to provide a list of genes of interest (genes.to.count parameter). One option in this case is to provide receptors involved in specific signaling whose abundance is to be illustrated on top of the number of interactions. For this, only.R.in.genes must be set to TRUE, the default. If not, then only interactions with both the ligand and the receptor in genes.to.count are considered. This may enable more specific counting. Lastly, it is possible to choose between color-coding the proportion of interactions with receptor or both receptor and ligand in genes.to.count, or to perform an enrichment analysis. In the last case, the color-code is based on -log10(P-values).

Value

A bubble plot is displayed reflecting population pairwise interactions.

Examples

```
print("cellNetBubblePlot")
if (FALSE) {
    cellNetBubblePlot(scsrcn)
}
```

---

cellNetHeatmap *Heatmap overview of cellular networks*

Description

Heatmap overview of cellular networks

Usage

```
cellNetHeatmap(
  obj,
  selected.populations = NULL,
  genes.to.count = NULL,
  only.R.in.genes = TRUE,
  use.proportions = FALSE,
  low.color = "white",
  high.color = "royalblue3",
```

```
  thres = NULL,
  col.fun = NULL,
  row.bar.color = "darkcyan",
  col.bar.color = "seagreen",
  pop.font.size = 10,
  title.font.size = 12,
  legend.font.size = 10,
  bar.plot.height = NULL,
  rect.border.width = 4,
  bar.num = FALSE,
  bar.num.font.size = 8,
  original.order = FALSE
)
```

Arguments

`obj`	A SCSRNet or SCSRNoNet object.
`selected.populations`	A vector of cell population names to consider in the plot. By default, all the populations are considered.
`genes.to.count`	A vector of gene names for counting or enrichment analysis.
`only.R.in.genes`	A logical indicating whether `genes.to.count` should be regarded as containing receptor gene names only.
`use.proportions`	A logical to choose between representing the proportion of genes in `genes.to.count` or its enrichment.
`low.color`	The color to be used for the lowest value.
`high.color`	The color to be used for the highest value.
`thres`	A higher threshold imposed to the values.
`col.fun`	A function that returns a color based on the values in the matrix. When no such function is provided, a function is generated from `low.color` and `high.color` with a linear gradient.
`row.bar.color`	The color of the barplot counting the total number of received LR interactions.
`col.bar.color`	The color of the barplot counting the total number of emitted LR interactions.
`pop.font.size`	Font size for cell population names.
`title.font.size`	Font size for row and column titles.
`legend.font.size`	Font size for the legends.
`bar.plot.height`	Heigh (or width) or the two barplots.
`rect.border.width`	White border width around the colored rectangles of the heatmap.
`bar.num`	A logical indicating whether numbers should be written on top of the barplots.
`bar.num.font.size`	Font size of the totals on the barplots.
`original.order`	A logical indicating whether matrix rows and columns should be left in their original order. If not (the default), they are reordered by computing two dendrograms that are not displayed.

Details

A heatmap is generated to represent how much each pair of cell populations interact including autocrine interactions when available.

By default, the plot only reports the number of interactions, but it is possible to provide a list of genes of interest (`genes.to.count` parameter). One option in this case is to provide receptors involved in specific signaling whose abundance is to be illustrated on top of the number of interactions. For this, `only.R.in.genes` must be set to TRUE, the default. If not, then only interactions with both the ligand and the receptor in `genes.to.count` are considered. This may enable more specific counting. Lastly, it is possible to choose between color-coding the proportion of interactions with receptor or both receptor and ligand in `genes.to.count`, or to perform an enrichment analysis. In the last case, the color-code is based on -log10(P-values).

Value

A heatmap is displayed reflecting population pairwise interactions.

Examples

```
print("cellNetHeatmap")
if (FALSE) {
    cellNetHeatmap(scsrcn)
}
```

---

example_dataset *An example of single-cell RNA-seq data set*

Description

A data set containing simulated read counts

Usage

```
data(example_dataset)
```

Format

A data frame with 1629 rows (genes) and 401 (cells)

---

getAutocrines *Method to retrieve autocrine interactions*

Description

Method to retrieve autocrine interactions

Method to retrieve autocrine interactions

Usage

```
## S4 method for signature 'SCSRNet'
getAutocrines(obj, pop)

## S4 method for signature 'SCSRNoNet'
getAutocrines(obj, pop)
```

Arguments

obj	A SCSRNoNet object.
pop	Name of the cell population whose autocrine ligand-receptor interactions should be retrieved.

Details

A BSRInferenceComp-class object is returned that contains all the inferred (unfiltered) interactions.

Interactions in tabular format can be obtained applying the LRinter accessor to the returned object. All the BSRInferenceComp methods to reduce pathway, ligand, or receptor redundancies can also be applied to the return object.

A data.frame is returned that contains the inferred interactions.

Value

A BSRInferenceComp object.

A data.frame.

Examples

```
print("getAutocrines")
data(example_dataset, package = "SingleCellSignalR")
mat <- log1p(data.matrix(example_dataset[, -1])) / log(2)
rownames(mat) <- example_dataset[[1]]
rme <- rowMeans(mat)
mmat <- mat[rme > 0.05, ]
d <- dist(t(mmat))
h <- hclust(d, method = "ward.D")
pop <- paste0("pop_", cutree(h, 5))
scsrcn <- SCSRNet(mat,
    normalize = FALSE, method = "log-only",
    log.transformed = TRUE, populations = pop
)
if(FALSE){
# infer ligand-receptor interactions from the comparison

scsrcn <- performInferences(scsrcn,
verbose = TRUE, min.logFC = 1e-10,
max.pval = 1)

getAutocrines(scsrcn, "pop_1")
}
print("getAutocrines")
data(example_dataset, package = "SingleCellSignalR")
mat <- log1p(data.matrix(example_dataset[, -1])) / log(2)
```

```
rownames(mat) <- example_dataset[[1]]
rme <- rowMeans(mat)
mmat <- mat[rme > 0.05, ]
d <- dist(t(mmat))
h <- hclust(d, method = "ward.D")
pop <- paste0("pop_", cutree(h, 5))
scsrnn <- SCSRNoNet(mat,
    normalize = FALSE, method = "log-only",
    log.transformed = TRUE, populations = pop
)
if(FALSE){
# infer ligand-receptor interactions from the comparison

scsrnn <- performInferences(scsrnn,
    verbose = TRUE, ,
    min.logFC = 1e-10, max.pval = 1)

getAutocrines(scsrcn, "pop_1")
}
```

`getParacrines` *Method to retrieve paracrine interactions*

Description

Method to retrieve paracrine interactions

Method to retrieve paracrine interactions

Usage

```
## S4 method for signature 'SCSRNet'
getParacrines(obj, source.pop, target.pop)

## S4 method for signature 'SCSRNoNet'
getParacrines(obj, source.pop, target.pop)
```

Arguments

`obj`	A SCSRNoNet object.
`source.pop`	Name of the cell population from which ligand-receptor interactions originate (express the ligands).
`target.pop`	Name of the cell population towards which ligand-receptor interactions go (express the receptors).

Details

A `BSRInferenceComp-class` object is returned that contains all the inferred (unfiltered) interactions.

Interactions in tabular format can be obtained applying the `LRinter` accessor to the returned object. All the BSRInferenceComp methods to reduce pathway, ligand, or receptor redundancies can also be applied to the return object.

A data.frame is returned that contains the inferred interactions.

Value

A BSRInferenceComp object.

A data.frame.

Examples

```
print("getParacrines")
data(example_dataset, package = "SingleCellSignalR")
mat <- log1p(data.matrix(example_dataset[, -1])) / log(2)
rownames(mat) <- example_dataset[[1]]
rme <- rowMeans(mat)
mmat <- mat[rme > 0.05, ]
d <- dist(t(mmat))
h <- hclust(d, method = "ward.D")
pop <- paste0("pop_", cutree(h, 5))
scsrcn <- SCSRNet(mat,
normalize = FALSE, method = "log-only",
log.transformed = TRUE, populations = pop
)
if(FALSE){
# infer ligand-receptor interactions from the comparison

scsrcn <- performInferences(scsrcn,
verbose = TRUE, min.logFC = 1e-10,
max.pval = 1)

getParacrines(scsrcn, "pop_1","pop_2")
}
print("getParacrines")
data(example_dataset, package = "SingleCellSignalR")
mat <- log1p(data.matrix(example_dataset[, -1])) / log(2)
rownames(mat) <- example_dataset[[1]]
rme <- rowMeans(mat)
mmat <- mat[rme > 0.05, ]
d <- dist(t(mmat))
h <- hclust(d, method = "ward.D")
pop <- paste0("pop_", cutree(h, 5))
scsrcn <- SCSRNoNet(mat,
normalize = FALSE, method = "log-only",
log.transformed = TRUE, populations = pop
)
if(FALSE){
# infer ligand-receptor interactions from the comparison

scsrcn <- performInferences(scsrcn,
    verbose = TRUE, ,
    min.logFC = 1e-10, max.pval = 1)

getParacrines(scsrcn, "pop_1","pop_2")
}
```

`infParamSC` *Inference parameters accessor*

Description

Inference parameters accessor

Inference parameters accessor

Usage

```
## S4 method for signature 'SCSRNet'
infParamSC(x)

## S4 method for signature 'SCSRNoNet'
infParamSC(x)
```

Arguments

x	SCSRNoNet object.

Value

infParamSC

Examples

```
print("infParamSC")
data(example_dataset, package = "SingleCellSignalR")
mat <- log1p(data.matrix(example_dataset[, -1])) / log(2)
rownames(mat) <- example_dataset[[1]]
rme <- rowMeans(mat)
mmat <- mat[rme > 0.05, ]
d <- dist(t(mmat))
h <- hclust(d, method = "ward.D")
pop <- paste0("pop_", cutree(h, 5))
scsrcn <- SCSRNet(mat,
    normalize = FALSE, method = "log-only",
    log.transformed = TRUE, populations = pop
)

infParamSC(scsrcn)
data(example_dataset, package = "SingleCellSignalR")
mat <- log1p(data.matrix(example_dataset[, -1])) / log(2)
rownames(mat) <- example_dataset[[1]]
rme <- rowMeans(mat)
mmat <- mat[rme > 0.05, ]
d <- dist(t(mmat))
h <- hclust(d, method = "ward.D")
pop <- paste0("pop_", cutree(h, 5))
scsrcn <- SCSRNoNet(mat,
normalize = FALSE, method = "log-only",
log.transformed = TRUE, populations = pop
)
if(FALSE){
# infer ligand-receptor interactions from the comparison

scsrcn <- performInferences(scsrcn,
    verbose = TRUE, ,
    min.logFC = 1e-10, max.pval = 1)
```

```
infParamSC(scsrcn)
}
```

`infParamSC<-,SCSRNet-method`
Inference parameters setter (internal use only)

Description

Inference parameters setter (internal use only)

Usage

```
## S4 replacement method for signature 'SCSRNet'
infParamSC(x) <- value
```

Arguments

`x`	SCSRNet object.
`value`	value to be set.

Value

returns `NULL`

`infParamSC<-,SCSRNoNet-method`
Inference parameters setter (internal use only)

Description

Inference parameters setter (internal use only)

Usage

```
## S4 replacement method for signature 'SCSRNoNet'
infParamSC(x) <- value
```

Arguments

`x`	SCSRNoNet object.
`value`	value to be set.

Value

returns `NULL`

paracrines *paracrines accessor*

Description

paracrines accessor

paracrines accessor

Usage

```
## S4 method for signature 'SCSRNet'
paracrines(x)

## S4 method for signature 'SCSRNoNet'
paracrines(x)
```

Arguments

x	SCSRNoNet object

Value

paracrines

paracrines

Examples

```
print("paracrines")
data(example_dataset, package = "SingleCellSignalR")
mat <- log1p(data.matrix(example_dataset[, -1])) / log(2)
rownames(mat) <- example_dataset[[1]]
rme <- rowMeans(mat)
mmat <- mat[rme > 0.05, ]
d <- dist(t(mmat))
h <- hclust(d, method = "ward.D")
pop <- paste0("pop_", cutree(h, 5))
scsrcn <- SCSRNet(mat,
    normalize = FALSE, method = "log-only",
    log.transformed = TRUE, populations = pop
)

paracrines(scsrcn)
```

paracrines<-,SCSRNet-method *paracrines setter (internal use only)*

Description

paracrines setter (internal use only)

Usage

```
## S4 replacement method for signature 'SCSRNet'
paracrines(x) <- value
```

Arguments

x	SCSRNet object
value	value to be set for paracrines

Value

returns NULL

paracrines<-,SCSRNoNet-method *paracrines setter (internal use only)*

Description

paracrines setter (internal use only)

Usage

```
## S4 replacement method for signature 'SCSRNoNet'
paracrines(x) <- value
```

Arguments

x	SCSRNoNet object
value	value to be set for paracrines

Value

returns NULL

performInferences *Inference of ligand-receptor interactions based on regulation*

Description

This method computes all the autocrine and paracrine ligand-receptor interactions. In addition, it is possible to restrict inferences to autocrine or paracrine only, or do it for chosen cell populations.

This method computes all the autocrine and paracrine ligand-receptor interactions. In addition, it is possible to restrict inferences to autocrine or paracrine only, or do it for chosen cell populations.

Usage

```
## S4 method for signature 'SCSRNet'
performInferences(
  obj,
  autocrine = TRUE,
  paracrine = TRUE,
  selected.populations = NULL,
  funDiffExpr = NULL,
  subsample.size = 50,
  n.resample = 50,
  check.pval.se = FALSE,
  rank.p = 0.55,
  max.pval = 0.05,
  min.logFC = 1,
  min.LR.score = 0,
  neg.receptors = FALSE,
  pos.targets = FALSE,
  neg.targets = FALSE,
  min.t.logFC = 0.5,
  restrict.genes = NULL,
  use.full.network = FALSE,
  reference = c("REACTOME-GOBP", "REACTOME", "GOBP"),
  max.pw.size = 600,
  min.pw.size = 10,
  min.positive = 2,
  restrict.pw = NULL,
  with.complex = TRUE,
 fdr.proc = c("BH", "Bonferroni", "Holm", "Hochberg", "SidakSS", "SidakSD", "BY", "ABH",
    "TSBH"),
  verbose = FALSE
)

## S4 method for signature 'SCSRNoNet'
performInferences(
  obj,
  autocrine = TRUE,
  paracrine = TRUE,
  selected.populations = NULL,
  funDiffExpr = NULL,
```

```
  subsample.size = 50,
  n.resample = 50,
  check.pval.se = FALSE,
  max.pval = 0.05,
  min.logFC = 1,
  min.LR.score = 0,
  neg.receptors = FALSE,
  restrict.genes = NULL,
 fdr.proc = c("BH", "Bonferroni", "Holm", "Hochberg", "SidakSS", "SidakSD", "BY", "ABH",
    "TSBH"),
  verbose = FALSE
)
```

Arguments

obj	A SCSRNoNet object.
autocrine	A logical indicating whether autocrine interactions should be inferred.
paracrine	A logical indicating whether paracrine interactions should be inferred.
selected.populations	A vector of cell population names to limit inferences to these very cell populations.
funDiffExpr	An optional function to compute the differential expression tables for each population. The function is called for one population at a time, with the SCSRNoNet object and the population name as parameters, and it must return a BSRClusterComp-class object.
subsample.size	The number of cells to sample from a given population to estimate differential expression significance. If set to -1, then all the cells of each population are used thus implying that the same observation from pairs of populations harboring different sizes will result in potentially very different P-values.
n.resample	The number of times the sampling is performed.
check.pval.se	A logical indicating that an estimate of the standard error on the differential expression P-value must be computed, and the conservative estimate P-value + SE must replace the P-value estimate.
rank.p	A number between 0 and 1 defining the rank of the last considered target genes (see BulkSignalR documentation).
max.pval	The maximum P-value imposed to both the ligand and the receptor.
min.logFC	The minimum log2 fold-change allowed for both the receptor and the ligand.
min.LR.score	The minimum required LR-score.
neg.receptors	A logical indicating whether receptors are only allowed to be upregulated (FALSE), or up- and downregulated (TRUE).
pos.targets	A logical imposing that all the network targets must display positive logFC, i.e. logFC >= min.t.logFC.
neg.targets	A logical imposing that all the network targets must display negative logFC, i.e. logFC <= - min.t.logFC.
min.t.logFC	The minimum log2 fold-change allowed for targets in case pos.targets or neg.targets are used.
restrict.genes	A list of gene symbols that restricts ligands and receptors.

`use.full.network`
: A logical to avoid limiting the reference network to the detected genes and use the whole reference network.

`reference`
: Which pathway reference should be used ("REACTOME" for Reactome, "GOBP" for GO Biological Process, or "REACTOME-GOBP" for both).

`max.pw.size`
: Maximum pathway size to consider from the pathway reference.

`min.pw.size`
: Minimum pathway size to consider from the pathway reference.

`min.positive`
: Minimum number of target genes to be found in a given pathway.

`restrict.pw`
: A list of pathway IDs to restrict the application of the function.

`with.complex`
: A logical indicating whether receptor co-complex members should be included in the target genes.

`fdr.proc`
: The procedure for adjusting P-values according to `mt.rawp2adjp`.

`verbose`
: A logical activating reports on computation steps done.

Details

The basis of the interaction inferences is the increased expression of the ligand, the receptor, and the target genes below the receptor in their respective cell populations. To determine differential expression it is necessary to compare cell populations and to generate data.frames representing gene differential expression. The data.frame format is defined in `BSRClusterComp-class` class. A default procedure is provided for generating all the differential expression analyses, i.e., one per cell population comparing it to all the other cells. It relies on Wilcoxon tests performed on a fixed number of cells per population (`subsample.size` parameter) to avoid biases due to actual sizes. Such sampling and significance analysis is performed `n.resample` times and the median P-value is finally used. It is possible to substitute a user-defined function for this purpose in case a different notion of differential expression would be preferred.

Once all the individual cell population differential analyses are done, autocrine and paracrine ligand-receptor interactions are generated.

In addition to statistical significance estimated according to BulkSignalR statistical model, we compute SingleCellSignalR original LR-score, based on L and R cluster average expression.

The basis of the interaction inferences is the increased expression of the ligand and the receptor in their respective cell populations. To determine differential expression it is necessary to compare cell populations and to generate data.frames representing gene differential expression. The data.frame format is defined in `BSRClusterComp-class` class. A default procedure is provided for generating all the differential expression analyses, i.e., one per cell population comparing it to all the other cells. It relies on Wilcoxon tests performed on a fixed number of cells per population (`subsample.size` parameter) to avoid biases due to actual sizes. Such sampling and significance analysis is performed `n.resample` times and the median P-value is finally used. It is possible to substitute a user-defined function for this purpose in case a different notion of differential expression would be preferred.

Once all the individual cell population differential analyses are done, autocrine and paracrine ligand-receptor interactions are generated.

In addition to statistical significance estimated by taking the ligand and the receptor P-values product, we compute SingleCellSignalR original LR-score, based on L and R cluster average expression.

Value

A SCSRNet object with inferences set.

A SCSRNoNet with inferences set.

Examples

```
# prepare data
print("SCSRNet")
data(example_dataset, package = "SingleCellSignalR")
mat <- log1p(data.matrix(example_dataset[, -1])) / log(2)
rownames(mat) <- example_dataset[[1]]
rme <- rowMeans(mat)
mmat <- mat[rme > 0.05, ]
d <- dist(t(mmat))
h <- hclust(d, method = "ward.D")
pop <- paste0("pop_", cutree(h, 5))
scsrcn <- SCSRNet(mat,
    normalize = FALSE, method = "log-only",
    log.transformed = TRUE, populations = pop
)

if (FALSE){
# infer ligand-receptor interactions from the comparison
scsrcn <- performInferences(scsrcn,
    verbose = TRUE, min.logFC = 1e-10,
    max.pval = 1
)
}
data(example_dataset, package = "SingleCellSignalR")
print("performInferences")
mat <- log1p(data.matrix(example_dataset[, -1])) / log(2)
rownames(mat) <- example_dataset[[1]]
rme <- rowMeans(mat)
mmat <- mat[rme > 0.05, ]
d <- dist(t(mmat))
h <- hclust(d, method = "ward.D")
pop <- paste0("pop_", cutree(h, 5))
scsrnn <- SCSRNoNet(mat,
    normalize = FALSE, method = "log-only",
    log.transformed = TRUE, populations = pop
)
# performInferences
if (FALSE) {
scsrnn <- performInferences(scsrnn,
    verbose = TRUE, ,
    min.logFC = 1e-10, max.pval = 1)
}
```

`populations` *populations accessor*

Description

populations accessor

populations accessor

Usage

```
## S4 method for signature 'SCSRNet'
populations(x)

## S4 method for signature 'SCSRNoNet'
populations(x)
```

Arguments

x	SCSRNoNet object

Value

populations

populations

Examples

```
print("populations")
data(example_dataset, package = "SingleCellSignalR")
mat <- log1p(data.matrix(example_dataset[, -1])) / log(2)
rownames(mat) <- example_dataset[[1]]
rme <- rowMeans(mat)
mmat <- mat[rme > 0.05, ]
d <- dist(t(mmat))
h <- hclust(d, method = "ward.D")
pop <- paste0("pop_", cutree(h, 5))
scsrcn <- SCSRNet(mat,
    normalize = FALSE, method = "log-only",
    log.transformed = TRUE, populations = pop
)

populations(scsrcn)
```

---

populations<-,SCSRNet-method

populations setter (internal use only)

Description

populations setter (internal use only)

Usage

```
## S4 replacement method for signature 'SCSRNet'
populations(x) <- value
```

Arguments

x	SCSRNet object
value	value to be set for populations

Value

returns NULL

`populations<-,SCSRNoNet-method` *populations setter (internal use only)*

Description

populations setter (internal use only)

Usage

```
## S4 replacement method for signature 'SCSRNoNet'
populations(x) <- value
```

Arguments

x	SCSRNoNet object
value	value to be set for populations

Value

returns NULL

`SCSRNet` *Instantiate a SCSRNet object from expression data*

Description

Take a matrix or a data frame containing single-cell RNA-sequencing or proteomics as input data and return a SCSRNet object ready for subsequent cellular network inference. Normally, SCSRNet objects are not instantiated directly, but through this function.

Usage

```
SCSRNet(
  counts,
  populations,
  normalize = TRUE,
  symbol.col = NULL,
  min.count = 1,
  prop = 0.1,
  method = c("UQ", "TC"),
  log.transformed = FALSE,
  min.LR.found = 80,
  species = "hsapiens",
  conversion.dict = NULL,
  UQ.pc = 0.95
)
```

Arguments

counts	A table or matrix of read counts.
populations	A vector indicating to which cell population each individual cell belongs to.
normalize	A logical indicating whether counts should be normalized according to method or if it was normalized beforehand.
symbol.col	The index of the column containing the gene symbols in case those are not the row names of counts already.
min.count	The minimum read count of a gene to be considered expressed in a sample.
prop	The minimum proportion of samples where a gene must be expressed higher than min.count to keep that gene.
method	The normalization method ('UQ' for upper quartile or 'TC' for total count). If normalize==FALSE, then method must be used to document the name of the normalization method applied by the user.
log.transformed	A logical indicating whether expression data were already log2-transformed.
min.LR.found	The minimum number of ligands or receptors found in count row names after eliminating the rows containing too many zeros according to min.count and prop.
species	Data were obtained for this organism.
conversion.dict	Correspondence table of HUGO gene symbols human/nonhuman. Not used unless the organism is different from human.
UQ.pc	Percentile for upper-quartile normalization, number between 0 and 1 (in case the default 0.95 is not appropriate).

Details

The counts matrix or table should be provided with expression levels of protein coding genes in each samples (column) and rownames(counts) set to HUGO official gene symbols. For commodity, it is also possible to provide counts with the gene symbols stored in one of its columns. This column must be specified with symbol.col. In such a case, prepareDataset will extract this column and use it to set the row names. Because row names must be unique, prepareDataset will eliminate rows with duplicated gene symbols by keeping the rows with maximum average expression. Gene symbol duplication may occur in protein coding genes after genome alignment due to errors in genome feature annotation files (GTF/GFF), where a handful of deprecated gene annotations might remain, or some genes are not given their fully specific symbols. If your read count extraction pipeline does not take care of this phenomenon, the maximum mean expression selection strategy implemented here should solve this difficulty for the sake of inferring ligand-receptor interactions.

If normalize is TRUE then normalization is performed according to method. If those two simple methods are not satisfying, then it is possible to provide a pre-normalized matrix setting normalize to FALSE. In such a case, the parameter method must be used to document the name of the normalization algorithm used.

In case proteomic data are provided, min.count must be understood as its equivalent with respect to those data.

Value

A SCSRNet object with empty interactions.

Examples

```
print("SCSRNet")
data(example_dataset, package = "SingleCellSignalR")
mat <- log1p(data.matrix(example_dataset[, -1])) / log(2)
rownames(mat) <- example_dataset[[1]]
rme <- rowMeans(mat)
mmat <- mat[rme > 0.05, ]
d <- dist(t(mmat))
h <- hclust(d, method = "ward.D")
pop <- paste0("pop_", cutree(h, 5))
scsrcn <- SCSRNet(mat,
    normalize = FALSE, method = "log-only",
    log.transformed = TRUE, populations = pop
)
```

SCSRNet-class *SingleCellSignalR cellular network object*

Description

An S4 class to represent data and inferences related to a single-cell data set and aimed at predicting ligand-receptor interactions. Both autocrine and paracrine interactions can be inferred and explored.

Details

This class is a container for all the data and inferences related to a given single-cell project. Inferred interactions in `paracrines` and `autocrines` can be further reduced to eliminate redundancies such as multiple downstream pathways for the same ligand-receptor interaction. See reduction functions `"reduceToBestPathway"`, `"reduceToLigand"`, `"reduceToReceptor"` and `"reduceToPathway"`.

Slots

`bsrdm.comp` A BulkSignalR BSRDataModelComp object containing the expression matrix data as well as the comparisons between cell populations.

`populations` A vector defining the cell population to which each individual cell belongs to. The length of this vector must be equal to the number of columns of the expression matrix.

`paracrines` A list of BulkSignalR BSRInferenceComp objects containing all the paracrine ligand-receptor interactions between all the possible pairs of cell populations.

`autocrines` A list of BulkSignalR BSRInferenceComp objects containing all the autocrine ligand-receptor interactions for each cell population.

`inf.param` Inference parameters.

Examples

```
print("Create SCSRNet object:")
data(example_dataset, package = "SingleCellSignalR")
mat <- log1p(data.matrix(example_dataset[, -1])) / log(2)
rownames(mat) <- example_dataset[[1]]
rme <- rowMeans(mat)
mmat <- mat[rme > 0.05, ]
```

```
d <- dist(t(mmat))
h <- hclust(d, method = "ward.D")
pop <- paste0("pop_", cutree(h, 5))
scsrcn <- SCSRNet(mat,
    populations = pop, normalize = FALSE,
    method = "log-only", log.transformed = TRUE
)
```

SCSRNoNet *Prepare a SCSRNoNet object from expression data*

Description

Take a matrix or data frame containing single-cell RNA sequencing or proteomics data and return a SCSRNoNet object ready for subsequent cellular network inference. Normally, SCSRNoNet objects are not instantiated directly, but through this function.

Usage

```
SCSRNoNet(
  counts,
  populations,
  normalize = TRUE,
  symbol.col = NULL,
  min.count = 1,
  prop = 0.1,
  method = c("UQ", "TC"),
  log.transformed = FALSE,
  min.LR.found = 80,
  species = "hsapiens",
  conversion.dict = NULL,
  UQ.pc = 0.95
)
```

Arguments

counts	A table or matrix of read counts.
populations	A vector indicating to which cell population each individual cell belongs to.
normalize	A logical indicating whether counts should be normalized according to method or if it was normalized beforehand.
symbol.col	The index of the column containing the gene symbols in case those are not the row names of counts already.
min.count	The minimum read count of a gene to be considered expressed in a sample.
prop	The minimum proportion of samples where a gene must be expressed higher than min.count to keep that gene.
method	The normalization method ('UQ' for upper quartile or 'TC' for total count). If normalize==FALSE, then method must be used to document the name of the normalization method applied by the user.
log.transformed	A logical indicating whether expression data were already log2-transformed.

`min.LR.found` The minimum number of ligands or receptors found in `count` row names after eliminating the rows containing too many zeros according to `min.count` and `prop`.

`species` Data were obtained for this organism.

`conversion.dict` Correspondence table of HUGO gene symbols human/nonhuman. Not used unless the organism is different from human.

`UQ.pc` Percentile for upper-quartile normalization, number between 0 and 1 (in case the default 0.95 is not appropriate).

Details

The `counts` matrix or table should be provided with expression levels of protein coding genes in each samples (column) and `rownames(counts)` set to HUGO official gene symbols. For commodity, it is also possible to provide `counts` with the gene symbols stored in one of its columns. This column must be specified with `symbol.col`. In such a case, `prepareDataset` will extract this column and use it to set the row names. Because row names must be unique, `prepareDataset` will eliminate rows with duplicated gene symbols by keeping the rows with maximum average expression. Gene symbol duplication may occur in protein coding genes after genome alignment due to errors in genome feature annotation files (GTF/GFF), where a handful of deprecated gene annotations might remain, or some genes are not given their fully specific symbols. If your read count extraction pipeline does not take care of this phenomenon, the maximum mean expression selection strategy implemented here should solve this difficulty for the sake of inferring ligand-receptor interactions.

If `normalize` is `TRUE` then normalization is performed according to `method`. If those two simple methods are not satisfying, then it is possible to provide a pre-normalized matrix setting `normalize` to `FALSE`. In such a case, the parameter `method` must be used to document the name of the normalization algorithm used.

In case proteomic data are provided, `min.count` must be understood as its equivalent with respect to those data.

Value

A SCSRNoNet object with empty interactions.

Examples

```
print("SCSRNoNet")
data(example_dataset, package = "SingleCellSignalR")
mat <- log1p(data.matrix(example_dataset[, -1])) / log(2)
rownames(mat) <- example_dataset[[1]]
rme <- rowMeans(mat)
mmat <- mat[rme > 0.05, ]
d <- dist(t(mmat))
h <- hclust(d, method = "ward.D")
pop <- paste0("pop_", cutree(h, 5))
scsrnn <- SCSRNoNet(mat,
    normalize = FALSE, method = "log-only",
    log.transformed = TRUE, populations = pop
)
```

SCSRNoNet-class *SingleCellSignalR no network object*

Description

An S4 class to represent data and naive, network-free inferences of ligand-receptor interactions. Both autocrine and paracrine interactions can be inferred and explored. The absence of receptor network downstream exploration makes computations fast, but the quality of the inferences is worse than those obtained using the SCSRNet class.

Details

This class is a container for all the data and inferences related to a given single-cell project.

Slots

`bsrdm.comp` A BulkSignalR BSRDataModelComp object containing the expression matrix data as well as the comparisons between cell populations.

`populations` A vector defining the cell population to which each individual cell belongs to. The length of this vector must be equal to the number of columns of the expression matrix.

`paracrines` A list of data.frames containing all the paracrine ligand-receptor interactions between all the possible pairs of cell populations.

`autocrines` A list of data.frames containing all the autocrine ligand-receptor interactions for each cell population.

`inf.param` Inference parameters.

Examples

```
print("Create SCSRNet object:")
data(example_dataset, package = "SingleCellSignalR")
mat <- log1p(data.matrix(example_dataset[, -1])) / log(2)
rownames(mat) <- example_dataset[[1]]
rme <- rowMeans(mat)
mmat <- mat[rme > 0.05, ]
d <- dist(t(mmat))
h <- hclust(d, method = "ward.D")
pop <- paste0("pop_", cutree(h, 5))
scsrnn <- SCSRNoNet(mat,
    populations = pop, normalize = FALSE,
    method = "log-only", log.transformed = TRUE
)
```

Index