Package: HyperCom (via r-universe)

November 22, 2026

Title A Hypergraph Method to Understand Cell-Cell Communication

Version 0.2.2

Description Infers cell-resolved cell-cell communication from single-cell transcriptomics data without aggregating cells by type (Moy and Przytycki (2026) <doi:10.64898/2026.09.30.755742>). HyperCom builds a hypergraph from known ligand-receptor interactions and the expression of those ligands and receptors in individual cells, diffuses over it by a random walk with restart, and scores each cell on a continuous spectrum from sender to receiver for each ligand-receptor pair. It then ranks ligand-receptor pairs by communication over the whole dataset or by condition, with optional permutation p-values. Input is a counts matrix with cell metadata, or a 'Seurat' or 'SingleCellExperiment' object; 'CellChat' ligand-receptor databases for human and mouse are included.

URL https://github.com/pawelab/HyperCom

BugReports https://github.com/pawelab/HyperCom/issues

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Imports dplyr (>= 1.1.0), Matrix, methods, parallelly (>= 1.23.0), Rcpp, stats, tidyr, utils

LinkingTo Rcpp

Suggests DelayedArray, ggplot2, ggridges, knitr, rmarkdown, SeuratObject, SingleCellExperiment, SummarizedExperiment, testthat (>= 3.1.7), withr

VignetteBuilder knitr

Config/testthat/edition 3

Depends R (>= 4.1)

LazyData true

License MIT + file LICENSE

Config/pak/sysreqs libicu-dev

Repository https://pawelab.r-universe.dev

Date/Publication 2026-11-22 16:55:50 UTC

RemoteUrl https://github.com/pawelab/HyperCom

RemoteRef HEAD

RemoteSha 7cbee2718fe5e04442c7270b7ffa7c16cb0f05cc

Contents

alexanian2021.counts . . . 2
alexanian2021.metadata . . . 3
cellchat.hs.lrdb . . . 4
cellchat.mm.lrdb . . . 4
filter_counts . . . 5
generate_adjacency_matrix . . . 6
generate_hyperedges . . . 7
generate_infMat . . . 8
generate_lrs_table . . . 9
generate_metadata . . . 10
HyperCom . . . 11
hypercom_adjacency . . . 13
hypercom_influence . . . 14
HyperCom_sce . . . 15
HyperCom_seurat . . . 17
prioritize_lr . . . 19
score_hypercom . . . 21

Index **23**

`alexanian2021.counts` *Example counts of 1000 mouse heart cells*

Description

Log-normalised expression of 1000 cells from mouse hearts after sham surgery or transverse aortic constriction (TAC), a subset of Alexanian et al. 2021, for trying HyperCom with `cellchat.mm.lrdb`.

Format

A sparse gene x cell matrix (dgCMatrix) of 17083 genes (mouse gene symbols) and 1000 cells (barcodes with the sample as suffix, e.g. "AACCGCGTCCAGAGGA-2"), in the order of the rows of `alexanian2021.metadata`

Author(s)

Michael Alexanian <michael.alexanian@gladstone.ucsf.edu>

References

https://doi.org/10.1038/s41586-021-03674-1

Examples

```
dim(alexanian2021.counts)
alexanian2021.counts[1:5, 1:3]
```

---

alexanian2021.metadata

Example metadata of 1000 mouse heart cells

Description

Cell metadata of `alexanian2021.counts`: 500 cells of a sham heart (sample 2) and 500 of a TAC heart (sample 3).

Format

A dataframe with row names being the cells of `alexanian2021.counts` and columns sample (integer), condition ("SHAM" or "TAC"), celltype (10 cell types) and UMAP_1, UMAP_2 (UMAP coordinates)

Author(s)

Michael Alexanian <michael.alexanian@gladstone.ucsf.edu>

References

https://doi.org/10.1038/s41586-021-03674-1

Examples

```
head(alexanian2021.metadata)
table(alexanian2021.metadata$sample, alexanian2021.metadata$condition)
```

cellchat.hs.lrdb *CellChat human ligand-receptor interactions*

Description

The ligand-receptor interactions of the CellChat database for H. sapiens, in the format HyperCom reads as `lrdb`.

Format

A dataframe of 2240 interactions with columns interaction (its name), ligand (a human gene symbol) and r1 to r4 (the receptor genes, NA where a receptor complex has fewer subunits)

Author(s)

Suoqin Jin <sqjin@whu.edu.cn>

References

https://doi.org/10.1038/s41467-021-21246-9

Examples

```
head(cellchat.hs.lrdb)
```

cellchat.mm.lrdb *CellChat mouse ligand-receptor interactions*

Description

The ligand-receptor interactions of the CellChat database for M. musculus, in the format HyperCom reads as `lrdb`.

Format

A dataframe of 2377 interactions with columns interaction (its name), ligand (a mouse gene symbol) and r1 to r4 (the receptor genes, NA where a receptor complex has fewer subunits)

Author(s)

Suoqin Jin <sqjin@whu.edu.cn>

References

https://doi.org/10.1038/s41467-021-21246-9

Examples

```
head(cellchat.mm.lrdb)
cellchat.mm.lrdb[cellchat.mm.lrdb$ligand == "Tgfb1", ]
```

`filter_counts` *Filter counts matrix to ligand and receptor genes*

Description

`filter_counts` returns the count matrix with only ligand and receptor genes

Usage

```
filter_counts(counts, lrdb, outfile = "0.counts.rds")
```

Arguments

`counts`	A matrix with row names being genes and column names being cells
`lrdb`	A dataframe of ligand-receptor interactions where the first column names the interaction, the second column is named ligand, and subsequent columns are named r1, ..., rn where n is the maximum number of genes in a receptor complex in the list of ligand-receptor interactions, e.g. `cellchat.mm.lrdb`
`outfile`	A string naming the output file, or NULL to write none

Details

This function takes a gene x cell counts matrix and filters it to just ligand and receptor genes with greater than 0 expression. Missing (NA) counts are ignored. A gene name used by several rows keeps its first row, with a warning. Raw and normalised counts give the same results (HyperCom uses whether each gene is above 0 in each cell); counts with negative values (scaled data) and counts with genes as column names stop.

Value

A matrix with row names being cells and column names being genes of only ligand and receptor genes with greater than 0 expression, sparse if counts is sparse

Examples

```
counts <- filter_counts(alexanian2021.counts, cellchat.mm.lrdb, outfile=NULL)
dim(counts)
```

`generate_adjacency_matrix` *Generate adjacency matrix of hypergraph*

Description

`generate_adjacency_matrix` returns the adjacency matrix of the hypergraph

Usage

```
generate_adjacency_matrix(hyperedges, outfile = "3.adj.rds")
```

Arguments

`hyperedges`	A dataframe of hyperedges from generate_hyperedges, with columns edge (the hyperedge) and node (a node in it)
`outfile`	A string naming the output file, or NULL to write none

Details

This function takes a dataframe of hyperedges and returns its adjacency matrix: each entry is the number of hyperedges two nodes share, and the diagonal is zero. Nodes are ordered by their names in the C locale, so the order does not depend on the session's collation. Nodes that share no hyperedge with another node are dropped. The full node x node matrix is not stored: the result records which nodes are in which hyperedge, so it takes memory in proportion to the hyperedge table instead of the number of nodes squared. It works like a matrix for dim, dimnames and rownames; `adj[i, j]` gives those entries and as.matrix the whole matrix.

Value

A hypercom_adjacency: a list with H, the sparse node x hyperedge matrix of how often each node is in each hyperedge

Examples

```
counts <- filter_counts(alexanian2021.counts, cellchat.mm.lrdb, outfile=NULL)
lrs <- generate_lrs_table(counts, cellchat.mm.lrdb, outfile=NULL)
hyperedges <- generate_hyperedges(counts, lrs, outfile=NULL)
adj <- generate_adjacency_matrix(hyperedges, outfile=NULL)
adj
dim(adj)
```

`generate_hyperedges` *Generate a table of hyperedges*

Description

`generate_hyperedges` returns a dataframe of hyperedges

Usage

```
generate_hyperedges(counts, lrs, outfile = "2.hyperedges.csv")
```

Arguments

`counts`	A filtered matrix with row names being cells and column names being genes, from filter_counts
`lrs`	A dataframe of ligand-receptor interactions from generate_lrs_table, in the format of lrdb with only interactions present in the counts matrix
`outfile`	A string naming the output file, or NULL to write none

Details

This function takes a filtered cell x gene counts matrix and table of ligand-receptor interactions to produce a table of hyperedges: one per ligand-receptor interaction (its genes) and one per gene (the gene and the cells that express it). Cells are nodes named by the row names of counts, so they must be unique (prefix barcodes with their sample when merging samples).

Value

A dataframe with one row per node of each hyperedge: columns edge (the hyperedge, an interaction or a gene), node (a gene or cell in it) and weight (1)

Examples

```
counts <- filter_counts(alexanian2021.counts, cellchat.mm.lrdb, outfile=NULL)
lrs <- generate_lrs_table(counts, cellchat.mm.lrdb, outfile=NULL)
hyperedges <- generate_hyperedges(counts, lrs, outfile=NULL)
head(hyperedges)
```

`generate_infMat` *Calculate influence matrix of hypergraph*

Description

`generate_infMat` returns the influence matrix of the hypergraph

Usage

```
generate_infMat(
  adj,
  r = 0.5,
  outfile = "4.infMat.rds",
  method = c("woodbury", "lu", "chol"),
  hyperedges = NULL,
  threads = NULL
)
```

Arguments

`adj`	An adjacency from generate_adjacency_matrix, or a named adjacency matrix
`r`	A double between 0 and 1 representing the restart probability
`outfile`	A string naming the output file, or NULL to write none
`method`	A string naming the method: "woodbury" (the default; "lu" for an adjacency matrix given without hyperedges), "lu" or "chol"
`hyperedges`	A dataframe of the hyperedges an adjacency matrix was made from, needed by method "woodbury" with an adjacency matrix; ignored with an adjacency from generate_adjacency_matrix, which holds its own
`threads`	An integer number of threads for method "woodbury", NULL for one per physical core; not used by "lu" and "chol"

Details

This function takes an adjacency matrix and returns its influence matrix by heat diffusion (a random walk with restart), r (I - (1 - r) adj / D)^-1 with D the row sums of adj. Three methods give the same matrix up to rounding (about 1e-15 relative, which can reorder exactly tied scores):

- "woodbury" (the default) works through the hyperedges, so no node x node matrix is made: memory grows with the number of nodes instead of its square. The result works like a matrix for dim, dimnames and rownames; `infMat[i, j]` gives those entries and as.matrix the whole matrix. An adjacency from generate_adjacency_matrix holds its hyperedges; an adjacency matrix needs `hyperedges`, and without them (and without a method given) gets "lu". Small data with at least as many hyperedges as nodes get the dense matrix of "chol", which is then no bigger.
- "lu" makes the dense matrix with exactly the numbers of HyperCom 0.0.0.9000; it holds two node x node matrices at once.

- "chol" makes the dense matrix faster and with one node x node matrix.

All methods stop when the matrix cannot be inverted (r near 0).

Value

An influence matrix with node names: a hypercom_influence (method "woodbury", unless there are at least as many hyperedges as nodes) or a dense matrix

Examples

```
counts <- filter_counts(alexanian2021.counts, cellchat.mm.lrdb, outfile=NULL)
lrs <- generate_lrs_table(counts, cellchat.mm.lrdb, outfile=NULL)
hyperedges <- generate_hyperedges(counts, lrs, outfile=NULL)
adj <- generate_adjacency_matrix(hyperedges, outfile=NULL)
infMat <- generate_infMat(adj, outfile=NULL)
infMat
infMat[c("Thbs1", "Sdc4"), 1:3]
```

---

generate_lrs_table *Generate a table of ligand-receptor interactions present in a counts matrix*

Description

`generate_lrs_table` returns a dataframe of ligand-receptor interactions present in a counts matrix

Usage

```
generate_lrs_table(counts, lrdb, outfile = "1.lrs.csv")
```

Arguments

`counts`	A filtered matrix with row names being cells and column names being genes, from filter_counts
`lrdb`	A dataframe of ligand-receptor interactions where the first column names the interaction, the second column is named ligand, and subsequent columns are named r1, ..., rn where n is the maximum number of genes in a receptor complex in the list of ligand-receptor interactions, e.g. `cellchat.mm.lrdb`
`outfile`	A string naming the output file, or NULL to write none

Details

This function takes a filtered cell x gene counts matrix and table of ligand-receptor interactions to produce a table of the ligand-receptor interactions whose genes are all in the counts matrix.

Value

A dataframe of ligand-receptor interactions in the format of lrdb with only interactions present in the counts matrix, and without receptor columns that are empty for all of them

Examples

```
counts <- filter_counts(alexanian2021.counts, cellchat.mm.lrdb, outfile=NULL)
lrs <- generate_lrs_table(counts, cellchat.mm.lrdb, outfile=NULL)
head(lrs)
```

---

generate_metadata *Generate cleaned metadata table*

Description

generate_metadata returns a minimal metadata dataframe

Usage

```
generate_metadata(
  infMat,
  metadata,
  sample,
  group = NULL,
  keep = NULL,
  outfile = "5.metadata.csv"
)
```

Arguments

infMat	An influence matrix from generate_infMat (only its node names are read)
metadata	A dataframe containing cell metadata with row names being cells
sample	A string that is the column name for sample in the provided metadata
group	A string that is the column name for group in the provided metadata
keep	A vector of strings that are additional column names to keep in the provided metadata
outfile	A string naming the output file, or NULL to write none

Details

This function takes an influence matrix and metadata table and creates a metadata data table containing the node name, sample, and group. If group is unspecified, it is assumed all nodes are part of the same group. Nodes of the influence matrix that are not cells of the metadata are genes: their sample, group and kept columns are "gene". Kept columns named cell, sample or group are renamed with make.unique (e.g. sample.1); kept dates and other classed columns other than factors become strings. Cells are compared only within their sample, so every sample needs at least 2 cells in the graph (cells that express a ligand or receptor gene).

Value

A dataframe with columns cell, sample, group and the kept columns, one row per node of the influence matrix

Examples

```
counts <- filter_counts(alexanian2021.counts, cellchat.mm.lrdb, outfile=NULL)
lrs <- generate_lrs_table(counts, cellchat.mm.lrdb, outfile=NULL)
hyperedges <- generate_hyperedges(counts, lrs, outfile=NULL)
adj <- generate_adjacency_matrix(hyperedges, outfile=NULL)
infMat <- generate_infMat(adj, outfile=NULL)
metadata <- generate_metadata(infMat, alexanian2021.metadata, sample="sample", group="condition",
                              outfile=NULL)
head(metadata)
table(metadata$sample)
```

HyperCom *Run the HyperCom pipeline*

Description

HyperCom runs the HyperCom pipeline

Usage

```
HyperCom(
  lrdb,
  counts,
  metadata,
  sample,
  group = NULL,
  keep = NULL,
  weights = c(1),
  significance = FALSE,
  parallel = FALSE,
  outdir = "HyperCom",
  threads = NULL,
  seed = NULL,
  verbose = TRUE,
  infmat.method = "woodbury",
  cache = TRUE
)
```

Arguments

lrdb	A dataframe of ligand-receptor interactions where the first column names the interaction, the second column is named ligand, and subsequent columns are named r1, ..., rn where n is the maximum number of genes in a receptor complex in the list of ligand-receptor interactions, e.g. cellchat.mm.lrdb

counts	A matrix with row names being genes and column names being cells
metadata	A dataframe containing cell metadata with row names being cells
sample	A string that is the column name for sample in the provided metadata
group	A string that is the column name for group in the provided metadata
keep	A vector of strings that are additional column names to keep in the provided metadata
weights	A vector of weights of at least 0, one result each, for the fraction of cells expressing a pair's genes in the priority score: 1 weighs it like the other four terms, less than 1 favors pairs of fewer cells, 0 leaves it out
significance	A boolean of whether to generate p-values for ligand-receptor interactions
parallel	A boolean of whether to run in parallel: one thread per physical core (one per logical core on Linux, where R does not tell them apart)
outdir	A string naming the output directory of the pipeline run
threads	An integer number of threads of at least 1, overriding parallel; more than the available cores warns and uses them all
seed	An integer seed for the background interactions of the p-values, NULL to use the current random state; the session's random state is restored afterwards
verbose	A boolean of whether to report progress as messages: what each step found, the pairs scored and the top pair of each weight
infmat.method	A string naming the method of generate_infMat: "woodbury" (the default; fastest and least memory), "chol" or "lu" (a full matrix of all cells and genes; "lu" gives exactly the numbers of HyperCom 0.0.0.9000). The methods agree to about 1e-15
cache	A boolean of whether to keep the pairs scored so far in <outdir>/cache, so that a stopped run continues where it stopped when run again; ignored when outdir is NULL

Details

This function runs filter_counts, generate_lrs_table, generate_hyperedges, generate_adjacency_matrix, generate_infMat, generate_metadata, and prioritize_lr, writing every step to outdir. By default no matrix with a row and a column for every cell and gene is made, so memory grows with the number of cells rather than its square (see generate_infMat). The arguments are checked before any step runs, and the run stops when no interaction of lrdb has all its genes expressed (e.g. lrdb of another species). Cells of counts that are not row names of metadata are left out, with a warning. Priority files of other weights already in outdir are kept, with a warning. If a run stops while pairs are scored, the same call run again repeats the earlier steps and continues the scoring from the cache in <outdir>/6.priority/cache (see prioritize_lr). The results are returned as well as written, so they need not be read back: res <- HyperCom(...), then res$priority[["1"]] or score_hypercom(res$infMat, res$metadata, ligand="Thbs1", receptors="Sdc4").

Value

Invisibly, a list of the results also written to outdir: priority (from prioritize_lr, one dataframe per weight), metadata (from generate_metadata), infMat (the influence matrix), adj (the adjacency) and lrs (the interactions with all genes expressed)

Examples

```
# a few ligands keep the example quick; use the whole database for real data
lrdb <- cellchat.mm.lrdb[cellchat.mm.lrdb$ligand %in% c("Thbs1", "Comp", "Tgfb1"), ]
res <- HyperCom(lrdb, alexanian2021.counts, alexanian2021.metadata, sample="sample",
                group="condition", outdir=file.path(tempdir(), "HyperCom"))
head(res$priority[["1"]])
list.files(file.path(tempdir(), "HyperCom"), recursive=TRUE)
scores <- score_hypercom(res$infMat, res$metadata, ligand="Thbs1", receptors="Sdc4")
```

---

hypercom_adjacency *Adjacency matrix of a hypergraph, stored as its incidence*

Description

hypercom_adjacency objects are made by generate_adjacency_matrix

Usage

```
## S3 method for class 'hypercom_adjacency'
dim(x)

## S3 method for class 'hypercom_adjacency'
dimnames(x)

## S3 method for class 'hypercom_adjacency'
x[i, j, drop = TRUE]

## S3 method for class 'hypercom_adjacency'
as.matrix(x, ...)

## S3 method for class 'hypercom_adjacency'
print(x, ...)
```

Arguments

x	A hypercom_adjacency
i, j	Indices of rows and columns
drop	A boolean of whether to drop dimensions of extent one
...	Ignored

Details

A hypercom_adjacency keeps the node x hyperedge incidence H of a hypergraph; its adjacency matrix is H H' with the diagonal set to zero. dim and dimnames give those of the matrix, adj[i, j] makes the indexed entries as a dense matrix (indices as for a matrix: names, positions or logical; factors are read by name, not by their codes) and as.matrix the whole dense matrix (nodes squared doubles).

Value

dim: the number of nodes twice; dimnames: the node names twice; [and as.matrix: a dense matrix; print: x, invisibly

Examples

```
counts <- filter_counts(alexanian2021.counts, cellchat.mm.lrdb, outfile=NULL)
lrs <- generate_lrs_table(counts, cellchat.mm.lrdb, outfile=NULL)
hyperedges <- generate_hyperedges(counts, lrs, outfile=NULL)
adj <- generate_adjacency_matrix(hyperedges, outfile=NULL)
adj
dim(adj)
adj[c("Thbs1", "Sdc4"), 1:3]
```

---

hypercom_influence *Influence matrix of a hypergraph, stored as its factors*

Description

hypercom_influence objects are made by generate_infMat with method "woodbury"

Usage

```
## S3 method for class 'hypercom_influence'
dim(x)

## S3 method for class 'hypercom_influence'
dimnames(x)

## S3 method for class 'hypercom_influence'
x[i, j, drop = TRUE]

## S3 method for class 'hypercom_influence'
as.matrix(x, threads = 1, ...)

## S3 method for class 'hypercom_influence'
print(x, ...)
```

Arguments

x	A hypercom_influence
i, j	Indices of rows and columns
drop	A boolean of whether to drop dimensions of extent one
threads	An integer number of threads for as.matrix
...	Ignored

Details

A hypercom_influence keeps the factors of the influence matrix F = r Delta^-1 D + r (1 - r) Y Z of a hypergraph (H, the node x hyperedge incidence; rs, the row sums D of the adjacency; delta, Delta = D + (1 - r) diag(H H'); Z, hyperedge x node; Y = H / Delta) instead of F. dim and dimnames give those of F, `infMat[i, j]` makes the indexed entries as a dense matrix (indices as for a matrix: names, positions or logical; factors are read by name, not by their codes) on one thread and as.matrix the whole dense matrix (nodes squared doubles).

Value

dim: the number of nodes twice; dimnames: the node names twice; [and as.matrix: a dense matrix; print: x, invisibly

Examples

```
counts <- filter_counts(alexanian2021.counts, cellchat.mm.lrdb, outfile=NULL)
lrs <- generate_lrs_table(counts, cellchat.mm.lrdb, outfile=NULL)
hyperedges <- generate_hyperedges(counts, lrs, outfile=NULL)
adj <- generate_adjacency_matrix(hyperedges, outfile=NULL)
infMat <- generate_infMat(adj, outfile=NULL)
infMat
dim(infMat)
infMat[c("Thbs1", "Sdc4"), 1:3]
```

HyperCom_sce *Run the HyperCom pipeline on a SingleCellExperiment object*

Description

`HyperCom_sce` runs the HyperCom pipeline on a SingleCellExperiment object

Usage

```
HyperCom_sce(
  sce,
  lrdb,
  sample,
  assay = "counts",
  group = NULL,
  keep = NULL,
  weights = c(1),
  significance = FALSE,
  parallel = FALSE,
  outdir = "HyperCom",
  threads = NULL,
  seed = NULL,
  verbose = TRUE,
```

```
  infmat.method = "woodbury",
  cache = TRUE
)
```

Arguments

sce	A SingleCellExperiment object
lrdb	A dataframe of ligand-receptor interactions where the first column names the interaction, the second column is named ligand, and subsequent columns are named r1, ..., rn where n is the maximum number of genes in a receptor complex in the list of ligand-receptor interactions, e.g. `cellchat.mm.lrdb`
sample	A string that is the column name for sample in the cell metadata (`colData`) of the SingleCellExperiment object
assay	A string naming the assay to read counts from ("counts" or "logcounts"; not a scaled assay)
group	A string that is the column name for group in the cell metadata of the SingleCellExperiment object
keep	A vector of strings that are additional column names to keep from the cell metadata of the SingleCellExperiment object
weights	A vector of weights of at least 0, one result each, for the fraction of cells expressing a pair's genes in the priority score: 1 weighs it like the other four terms, less than 1 favors pairs of fewer cells, 0 leaves it out
significance	A boolean of whether to generate p-values for ligand-receptor interactions
parallel	A boolean of whether to run in parallel: one thread per physical core (one per logical core on Linux, where R does not tell them apart)
outdir	A string naming the output directory of the pipeline run
threads	An integer number of threads of at least 1, overriding parallel; more than the available cores warns and uses them all
seed	An integer seed for the background interactions of the p-values, NULL to use the current random state; the session's random state is restored afterwards
verbose	A boolean of whether to report progress as messages: what each step found, the pairs scored and the top pair of each weight
infmat.method	A string naming the method of `generate_infMat`: "woodbury" (the default; fastest and least memory), "chol" or "lu" (a full matrix of all cells and genes; "lu" gives exactly the numbers of HyperCom 0.0.0.9000). The methods agree to about 1e-15
cache	A boolean of whether to keep the pairs scored so far in `<outdir>/cache`, so that a stopped run continues where it stopped when run again; ignored when outdir is NULL

Details

This function runs `HyperCom` on one assay and the cell metadata (`colData`) of a SingleCellExperiment, or of any SummarizedExperiment with genes as rows and cells as columns. The row names must be gene symbols and the column names cell names. Assays held on disk (DelayedArray, HDF5Array) are read for the genes of lrdb only. Needs the SummarizedExperiment package.

Value

Invisibly, a list of the results also written to outdir: priority (from prioritize_lr, one dataframe per weight), metadata (from generate_metadata), infMat (the influence matrix), adj (the adjacency) and lrs (the interactions with all genes expressed)

Examples

```
sce <- SingleCellExperiment::SingleCellExperiment(assays=list(counts=alexanian2021.counts),
                                                  colData=alexanian2021.metadata)
lrdb <- cellchat.mm.lrdb[cellchat.mm.lrdb$ligand %in% c("Thbs1", "Comp", "Tgfb1"), ]
res <- HyperCom_sce(sce, lrdb, sample="sample", group="condition",
                    outdir=file.path(tempdir(), "HyperCom_sce"))
head(res$priority[["1"]])
```

HyperCom_seurat *Run the HyperCom pipeline on a seurat object*

Description

`HyperCom_seurat` runs the HyperCom pipeline on a Seurat object

Usage

```
HyperCom_seurat(
  seurat,
  lrdb,
  sample,
  assay = "RNA",
  layer = "counts",
  group = NULL,
  keep = NULL,
  weights = c(1),
  significance = FALSE,
  parallel = FALSE,
  outdir = "HyperCom",
  threads = NULL,
  seed = NULL,
  verbose = TRUE,
  infmat.method = "woodbury",
  cache = TRUE
)
```

Arguments

`seurat`	A Seurat object

lrdb	A dataframe of ligand-receptor interactions where the first column names the interaction, the second column is named ligand, and subsequent columns are named r1, ..., rn where n is the maximum number of genes in a receptor complex in the list of ligand-receptor interactions, e.g. cellchat.mm.lrdb
sample	A string that is the column name for sample in the cell metadata of the Seurat object
assay	A string naming the assay to read counts from
layer	A string naming the layer of the assay to read counts from
group	A string that is the column name for group in the cell metadata of the Seurat object
keep	A vector of strings that are additional column names to keep from the cell metadata of the Seurat object
weights	A vector of weights of at least 0, one result each, for the fraction of cells expressing a pair's genes in the priority score: 1 weighs it like the other four terms, less than 1 favors pairs of fewer cells, 0 leaves it out
significance	A boolean of whether to generate p-values for ligand-receptor interactions
parallel	A boolean of whether to run in parallel: one thread per physical core (one per logical core on Linux, where R does not tell them apart)
outdir	A string naming the output directory of the pipeline run
threads	An integer number of threads of at least 1, overriding parallel; more than the available cores warns and uses them all
seed	An integer seed for the background interactions of the p-values, NULL to use the current random state; the session's random state is restored afterwards
verbose	A boolean of whether to report progress as messages: what each step found, the pairs scored and the top pair of each weight
infmat.method	A string naming the method of generate_infMat: "woodbury" (the default; fastest and least memory), "chol" or "lu" (a full matrix of all cells and genes; "lu" gives exactly the numbers of HyperCom 0.0.0.9000). The methods agree to about 1e-15
cache	A boolean of whether to keep the pairs scored so far in <outdir>/cache, so that a stopped run continues where it stopped when run again; ignored when outdir is NULL

Details

This function runs HyperCom on one assay layer and the cell metadata of a Seurat object (v5 or older assays). Needs the SeuratObject package.

Value

Invisibly, a list of the results also written to outdir: priority (from prioritize_lr, one dataframe per weight), metadata (from generate_metadata), infMat (the influence matrix), adj (the adjacency) and lrs (the interactions with all genes expressed)

Examples

```
seurat <- SeuratObject::CreateSeuratObject(alexanian2021.counts, meta.data=alexanian2021.metadata)
lrdb <- cellchat.mm.lrdb[cellchat.mm.lrdb$ligand %in% c("Thbs1", "Comp", "Tgfb1"), ]
res <- HyperCom_seurat(seurat, lrdb, sample="sample", group="condition",
                       outdir=file.path(tempdir(), "HyperCom_seurat"))
head(res$priority[["1"]])
```

`prioritize_lr` *Order ligand-receptor pairs by signaling strength and specificity*

Description

`prioritize_lr` returns an ordered list of ligand-receptor pairs

Usage

```
prioritize_lr(
  lrs,
  adj,
  infMat,
  metadata,
  weights = c(1),
  significance = FALSE,
  parallel = FALSE,
  outdir = "6.priority",
  threads = NULL,
  seed = NULL,
  verbose = TRUE,
  cache = TRUE
)
```

Arguments

`lrs`	A dataframe of ligand-receptor interactions from generate_lrs_table, in the format of lrdb with only interactions present in the counts matrix
`adj`	An adjacency from generate_adjacency_matrix, or a named adjacency matrix
`infMat`	An influence matrix from generate_infMat (a hypercom_influence or a named dense matrix)
`metadata`	A dataframe from generate_metadata, containing columns named cell, sample and group
`weights`	A vector of weights of at least 0, one result each, for the fraction of cells expressing a pair's genes in the priority score: 1 weighs it like the other four terms, less than 1 favors pairs of fewer cells, 0 leaves it out
`significance`	A boolean of whether to generate p-values for ligand-receptor interactions

parallel	A boolean of whether to run in parallel: one thread per physical core (one per logical core on Linux, where R does not tell them apart)
outdir	A string naming the output directory, or NULL to write no files
threads	An integer number of threads of at least 1, overriding parallel; more than the available cores warns and uses them all
seed	An integer seed for the background interactions of the p-values, NULL to use the current random state; the session's random state is restored afterwards
verbose	A boolean of whether to report progress as messages: the pairs scored about every 5 s with the time left, and the top pair of each weight
cache	A boolean of whether to keep the pairs scored so far in <outdir>/cache, so that a stopped run continues where it stopped when run again; ignored when outdir is NULL

Details

This function takes the ligand-receptor pairs, the adjacency matrix, a named influence matrix and the corresponding metadata and ranks the pairs by a priority score: the weighted geometric mean of five terms of the cells that express a pair's genes (their fraction of all cells, the smallest fraction of cells expressing one of its genes, the mean of each cell's larger score as sender or receiver, how evenly they split into senders and receivers, and how much they concentrate in some groups). Cells are scored once for all weights; parallel or threads use several CPU cores. With significance, the p-value of a pair is the fraction of 1000 background pairs of random genes whose priority is at least as high, so it moves in steps of 0.001 and is 0 when no background pair scored as high; p.adj is its Benjamini-Hochberg adjustment. Background pairs without a priority (NaN, e.g. when some of their cells have no score) are left out, with a warning. Background pairs are first screened: one pass over their cell grids bounds their priorities without ranking the cells, and only the few whose bounds hold a real pair's priority are scored in full, so the p-values are those of scoring all of them, in less time. With outdir set and cache=TRUE, the pairs scored so far are kept in <outdir>/cache while pairs are scored, and the folder is removed once the priority files are written. If the run stops (time limit of a cluster job, crash, Esc), the same call run again continues where it stopped, losing at most the last batch of pairs, with the same results as a run that never stopped on the same type of computer (on another type the BLAS can change the last bits of a "woodbury" influence matrix, and priorities by about 1e-13); without seed it also continues with the background pairs drawn before. Weights and threads may differ. A cache made from other inputs (or damaged) is not used: it is moved to <outdir>/cache.old, with a warning. If the cache cannot be written, scoring goes on without it, with a warning.

Value

A list of dataframes named by weight, each the ligand-receptor pairs ordered by their priority column, with columns p and p.adj when significance is TRUE

Examples

```
counts <- filter_counts(alexanian2021.counts, cellchat.mm.lrdb, outfile=NULL)
lrs <- generate_lrs_table(counts, cellchat.mm.lrdb, outfile=NULL)
hyperedges <- generate_hyperedges(counts, lrs, outfile=NULL)
adj <- generate_adjacency_matrix(hyperedges, outfile=NULL)
```

```
infMat <- generate_infMat(adj, outfile=NULL)
metadata <- generate_metadata(infMat, alexanian2021.metadata, sample="sample", group="condition",
                              outfile=NULL)
# a few ligands keep the example quick; significance=TRUE adds p-values from 1000 background pairs
pairs <- lrs[lrs$ligand %in% c("Thbs1", "Comp", "Tgfb1"), ]
priority <- prioritize_lr(pairs, adj, infMat, metadata, weights=c(1, 0.5), outdir=NULL)
head(priority[["1"]])
```

`score_hypercom` *Calculate the HyperCom score for each cell*

Description

`score_hypercom` returns a dataframe containing the HyperCom score for each cell

Usage

```
score_hypercom(
  infMat,
  metadata,
  ligand = NULL,
  receptors = NULL,
  parallel = FALSE,
  threads = NULL
)
```

Arguments

`infMat`	An influence matrix from generate_infMat (a hypercom_influence or a named dense matrix)
`metadata`	A dataframe from generate_metadata, containing columns named cell, sample and group
`ligand`	A string of the ligand name, NULL for the network score
`receptors`	A vector containing string elements of the receptor names, NULL for the network score
`parallel`	A boolean of whether to run in parallel: one thread per physical core (one per logical core on Linux, where R does not tell them apart)
`threads`	An integer number of threads of at least 1, overriding parallel; more than the available cores warns and uses them all

Details

This function takes a named influence matrix, the corresponding metadata, the ligand, and receptor to give each cell a HyperCom score. With a ligand and receptors this is the ligand-receptor pair score; with neither, the network score. For a receptor complex the scores of its subunits are averaged. Scores are the same at any number of threads.

Value

A dataframe with one row per cell, sorted by hypercom (highest first), with columns cell, sender and receiver (the cell's sender and receiver scores), dif (receiver - sender), avg (their mean), max (the larger of the two) and hypercom (the HyperCom score: the sign of dif times the percent rank of its size times avg)

Examples

```
counts <- filter_counts(alexanian2021.counts, cellchat.mm.lrdb, outfile=NULL)
lrs <- generate_lrs_table(counts, cellchat.mm.lrdb, outfile=NULL)
hyperedges <- generate_hyperedges(counts, lrs, outfile=NULL)
adj <- generate_adjacency_matrix(hyperedges, outfile=NULL)
infMat <- generate_infMat(adj, outfile=NULL)
metadata <- generate_metadata(infMat, alexanian2021.metadata, sample="sample", group="condition",
                              outfile=NULL)
scores <- score_hypercom(infMat, metadata, ligand="Thbs1", receptors="Sdc4")
head(scores)
# a receptor complex: the scores of its subunits are averaged
complex <- score_hypercom(infMat, metadata, ligand="Tgfb1", receptors=c("Tgfbr1", "Tgfbr2"))
# the network score: no ligand and no receptors
network <- score_hypercom(infMat, metadata)
```

Index

∗ **data**
    alexanian2021.counts, 2
    alexanian2021.metadata, 3
    cellchat.hs.lrdb, 4
    cellchat.mm.lrdb, 4
[.hypercom_adjacency
        *(*hypercom_adjacency*)*, 13
[.hypercom_influence
        *(*hypercom_influence*)*, 14

alexanian2021.counts, 2
alexanian2021.metadata, 3
as.matrix.hypercom_adjacency
        *(*hypercom_adjacency*)*, 13
as.matrix.hypercom_influence
        *(*hypercom_influence*)*, 14

cellchat.hs.lrdb, 4
cellchat.mm.lrdb, 4

dim.hypercom_adjacency
        *(*hypercom_adjacency*)*, 13
dim.hypercom_influence
        *(*hypercom_influence*)*, 14
dimnames.hypercom_adjacency
        *(*hypercom_adjacency*)*, 13
dimnames.hypercom_influence
        *(*hypercom_influence*)*, 14

filter_counts, 5

generate_adjacency_matrix, 6
generate_hyperedges, 7
generate_infMat, 8
generate_lrs_table, 9
generate_metadata, 10

HyperCom, 11
hypercom_adjacency, 13
hypercom_influence, 14
HyperCom_sce, 15
HyperCom_seurat, 17

print.hypercom_adjacency
        *(*hypercom_adjacency*)*, 13
print.hypercom_influence
        *(*hypercom_influence*)*, 14
prioritize_lr, 19

score_hypercom, 21