Package: blisa (via r-universe)

July 10, 2026

Type Package

Title Infer Cell-Cell Communication from Spatial Transcriptomics

Version 1.0.0

Description Identifies cell-cell communication hotspots in spatial transcriptomics data using bivariate Local Moran's I statistics on hexagonally binned cells. Provides functions for spatial weighting, ligand-receptor pair filtering, hotspot detection, and visualisation of sender-receiver cell-type interactions.

License GPL (>= 3)

URL https://github.com/ChenLaboratory/blisa, https://chenlaboratory.github.io/blisa/

BugReports https://github.com/ChenLaboratory/blisa/issues

Encoding UTF-8

LazyData false

Imports sf, spdep, fastLISA, Matrix, SpatialExperiment, SummarizedExperiment, ComplexHeatmap, ggplot2, viridisLite, grid

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

VignetteBuilder knitr

Suggests knitr, rmarkdown

Config/pak/sysreqs libabsl-dev cmake libgdal-dev gdal-bin libgeos-dev libmagick++-dev gsfonts libicu-dev libpng-dev libssl-dev perl libproj-dev libsqlite3-dev libudunits2-dev zlib1g-dev

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

Date/Publication 2026-07-10 00:16:30 UTC

RemoteUrl https://github.com/chenlaboratory/blisa

RemoteRef HEAD

RemoteSha 2a23260da7646e1b575beb5f27bdb40d22a3fa35

Contents

`blisa-package` *blisa: Bivariate Local Indicator of Spatial Association for spatial transcriptomics*

Description

Identifies cell-cell communication hotspots in spatial transcriptomics data using bivariate Local Moran's I statistics on hexagonally binned cells. Provides functions for spatial weighting, ligand-receptor pair filtering, hotspot detection, and visualisation of sender-receiver cell-type interactions.

Author(s)

Maintainer: Yunshun Chen <yuchen@wehi.edu.au>

Authors:

- Lei Qin <qin.l@wehi.edu.au>
- Lizhong Chen

See Also

Useful links:

- https://github.com/ChenLaboratory/blisa
- https://chenlaboratory.github.io/blisa/
- Report bugs at https://github.com/ChenLaboratory/blisa/issues

`blisa` *Run BLISA spatial cell-cell communication analysis*

Description

Generic function for running BLISA (Bivariate Local Indicator of Spatial Association). Dispatches on the class of `x`:

- `blisa.default` accepts a pre-binned gene-by-bin count matrix and a matching `bins` polygon object.
- `blisa.SpatialExperiment` accepts a cell-level `SpatialExperiment` object and bins cells into hexagonal tiles internally via `hexBinCells` before running the analysis.

Usage

```
blisa(x, ...)

## Default S3 method:
blisa(
  x,
  bins,
  LR_df = NULL,
  bin_size = 50,
  dmax = 250,
  nsim = 999,
  p_cutoff = 0.05,
  min_ligand = 10,
  min_receptor = 10,
  min_cells = 1,
  n_cells_col = NA,
  annotation_col = "annotation",
  default_mode = "diffuse",
  diffuse_category = c("Secreted Signaling", "Non-protein Signaling"),
  species = c("human", "mouse"),
  genes = NULL,
  counts_by_group = NULL,
  fast = TRUE,
  cpu_threads = 4L,
  verbose = FALSE,
  ...
)

## S3 method for class 'SpatialExperiment'
blisa(
  x,
  bin_size = 50,
  LR_df = NULL,
```

```
  group = "cell_type",
  genes = NULL,
  min_cells = 1,
  min_total_counts = 10,
  verbose = FALSE,
  ...
)
```

Arguments

x	A gene-by-bin count matrix (for `blisa.default`) or a cell-level `SpatialExperiment` object (for `blisa.SpatialExperiment`).
...	Additional arguments passed to the relevant method.
bins	An `sf` object of bin polygons. Row order must match the column order of `x`.
LR_df	Data frame of ligand-receptor pairs with columns `ligand.symbol` and `receptor.symbol`. When `NULL`, CellChatDB for the chosen `species` is downloaded automatically.
bin_size	Numeric. Width of each hexagonal bin in coordinate units (e.g. microns). Passed to `hexBinCells` and `computeSpatialWeights`. Default `50`.
dmax	Numeric. Maximum distance for diffuse-mode neighbours. Default `250`.
nsim	Integer. Number of permutations for Moran's I significance. Default `999`.
p_cutoff	Numeric. P-value threshold for High-High hotspots. Default `0.05`.
min_ligand	Numeric. Minimum ligand count threshold. Default `10`.
min_receptor	Numeric. Minimum receptor count threshold. Default `10`.
min_cells	Integer. Bins with fewer cells are dropped during binning by `hexBinCells`. Default `1`.
n_cells_col	Character or `NA`. Column in `bins` holding per-bin cell counts used for `min_cells` filtering. Set to `NA` to skip (default).
annotation_col	Character. Column in `LR_df` specifying interaction category used for communication-mode assignment. Default `"annotation"`.
default_mode	Character. CCC mode assigned to LR pairs whose annotation does not match `diffuse_category`. Default `"diffuse"`.
diffuse_category	Character vector of annotation categories treated as diffuse signalling.
species	Character. Which CellChatDB to download when `LR_df = NULL`. One of `"human"` (default) or `"mouse"`.
genes	Character vector of gene names to consider when matching ligand-receptor pairs. Defaults to `rownames(x)` (all genes in the `SpatialExperiment` object).
counts_by_group	Named list of gene-by-bin count matrices, one per group level (e.g. cell type), as returned by `hexBinCells` when `group` is supplied. When provided, `runCCI` is called automatically after the BLISA loop and its output is included in the result as `CCI_scores`. Default `NULL`.

`fast` Logical. When `TRUE` (default), uses `fastLISA::local_moran_bv` (a fast C/OpenMP backend) for the bivariate local Moran's I computation. When `FALSE`, uses the original `spdep::localmoran_bv` + `spdep::hotspot` pipeline.

`cpu_threads` Integer. Number of OpenMP threads used by `fastLISA::local_moran_bv`. Only used when `fast = TRUE`. Ignored on platforms without OpenMP. Default `4L`.

`verbose` Logical. If `TRUE`, print progress messages. Default `FALSE` (silent).

`group` Character. Column name in `colData(x)` to use as the grouping variable (e.g. cell type) for per-group bin aggregation and downstream CCI analysis via `runCCI`. If the column is not found in `colData(x)`, a message is issued and CCI is skipped. Default `"cell_type"`.

`min_total_counts` Numeric. Bins whose total counts (summed over all genes) fall below this threshold are dropped during binning by `hexBinCells`. Set to `0` to disable. Default `10`.

Value

A list; see individual method documentation for details.

An object of class `blisa` with four components:

LR_results Data frame of BLISA results for each LR pair, including `ccc_mode`, `sig_numbers`, `sig_index`, `sig_pval`, `all_pval`, `all_lisa`, `all_quadrant`, and original columns from `LR_df`. `all_quadrant` is a character vector of hotspot quadrant labels (`"High-High"`, `"Low-Low"`, etc.) for every bin; non-tested bins are `NA`.

bins Bin-level `sf` object of hexagonal polygons.

spatial_weights Spatial weights list from `computeSpatialWeights`.

CCI_scores `NULL` unless `counts_by_group` is supplied, in which case a wide data frame of interaction scores from `runCCI`: rows are `"Sender->Receiver"` group pairs, columns are LR pairs.

Methods (by class)

- `blisa(default)`: Method for a gene-by-bin count matrix.
- `blisa(SpatialExperiment)`: Method for a cell-level SpatialExperiment object. Bins cells into hexagonal tiles via `hexBinCells` then delegates to `blisa.default`.

Examples

```
## Not run:
# Download and cache the example Xenium breast cancer dataset (~21 MB)
data_url  <- paste0(
  "https://github.com/ChenLaboratory/example_data/releases/",
  "download/v1.0.0/spe_xenium_bc_s1rep1.rds"
)
cache_dir <- tools::R_user_dir("blisa", "cache")
data_file <- file.path(cache_dir, "spe_xenium_bc_s1rep1.rds")
```

```
if (!file.exists(data_file)) {
  dir.create(cache_dir, recursive = TRUE, showWarnings = FALSE)
  download.file(data_url, data_file, mode = "wb")
}
spe <- readRDS(data_file)

# blisa.SpatialExperiment method: bins cells, runs LISA, scores CCI
result <- blisa(spe, bin_size = 50, group = "cell_type")
result

## End(Not run)
```

`computeSpatialWeights` *Compute Spatial Weights for BLISA*

Description

Builds queen (nearby) and distance-decay (diffuse) spatial weight matrices from a bin-level `sf` object, excluding isolated bins and optionally excluding low-cell bins. A second-pass isolation check further removes bins that become isolated after the initial subset.

Usage

```
computeSpatialWeights(
  bins,
  bin_size = 50,
  dmax = 250,
  min_cells = 1,
  n_cells_col = NA
)
```

Arguments

`bins`	An `sf` object of spatial bins.
`bin_size`	Numeric. Bin spacing used to define queen adjacency (`1.2 * bin_size` radius).
`dmax`	Numeric. Maximum distance for diffuse-mode neighbours.
`min_cells`	Integer. Minimum cell count for a bin to be included. Ignored when `n_cells_col = NA`.
`n_cells_col`	Character or `NA`. Column name in `bins` holding per-bin cell counts. Set to `NA` to skip cell-count filtering (default).

Value

A list with:

queen_wt Spatial weights list for nearby (queen) mode.

dist_wt Spatial weights list for diffuse (distance-decay) mode.

keep_idx_queen Integer indices of bins used in queen-mode Moran.

keep_idx_dist Integer indices of bins used in diffuse-mode Moran.

isolate_idx_queen Integer indices of original queen-mode isolates.

isolate_idx_dist Integer indices of original diffuse-mode isolates.

low_cell_idx Integer indices of bins excluded for low cell counts.

queen_nb_full Full (unsubset) neighbour list for nearby mode, indexed over all bins.

dist_nb_full Full (unsubset) neighbour list for diffuse mode, indexed over all bins.

Examples

```
## Not run:
set.seed(42)
pts  <- sf::st_as_sf(
  data.frame(x = runif(300, 0, 1000), y = runif(300, 0, 1000)),
  coords = c("x", "y"), crs = NA
)
bins <- sf::st_sf(
  geometry = sf::st_make_grid(pts, cellsize = 100,
                              what = "polygons", square = FALSE)
)
sw <- computeSpatialWeights(bins, bin_size = 100, dmax = 300)
names(sw)

## End(Not run)
```

`filterLRpairs` *Filter ligand-receptor pairs by expression threshold*

Description

Retains only LR pairs where at least one bin/spot has counts at or above `min_ligand` for every ligand subunit and `min_receptor` for every receptor subunit.

Usage

```
filterLRpairs(
  counts,
  min_ligand = 10,
  min_receptor = 10,
  LR_df = NULL,
  species = c("human", "mouse")
)
```

Arguments

counts	Gene-by-bin count matrix (dense or sparse). Row names must be gene symbols.
min_ligand	Numeric. Minimum count threshold for ligand genes. At least one bin must meet or exceed this value. Default 10.
min_receptor	Numeric. Minimum count threshold for receptor genes. At least one bin must meet or exceed this value. Default 10.
LR_df	Data frame of ligand-receptor pairs with columns ligand.symbol and receptor.symbol (comma-separated gene symbols for multi-subunit complexes). When NULL, the CellChatDB for the chosen species is downloaded automatically.
species	Character. Which CellChatDB to download when LR_df is NULL. One of "human" (default) or "mouse".

Value

A subset of LR_df containing only pairs that pass the expression thresholds for both ligand and receptor.

Examples

```
## Not run:
# Supply a small custom LR_df to avoid a network download
LR_df <- data.frame(
  ligand.symbol   = c("GENE1", "GENE3"),
  receptor.symbol = c("GENE2", "GENE4"),
  annotation      = c("Secreted Signaling", "ECM-Receptor"),
  row.names       = c("LR1", "LR2")
)
set.seed(1)
counts <- matrix(
  rpois(4 * 50, lambda = c(20, 1, 5, 20)), nrow = 4, ncol = 50,
  dimnames = list(c("GENE1", "GENE2", "GENE3", "GENE4"),
                  paste0("bin_", 1:50))
)
filterLRpairs(counts, min_ligand = 10, min_receptor = 10, LR_df = LR_df)

## End(Not run)
```

---

hexBinCells *Bin cells into hexagonal spatial bins*

Description

Aggregates single-cell spatial data into hexagonal bins and returns a bin-level count matrix together with a matching sf polygon object, ready to pass directly to blisa.default.

Usage

```
hexBinCells(
  coords_df,
  counts_matrix,
  bin_size = 50,
  min_cells = 1,
  min_total_counts = 10,
  group = NULL,
  verbose = FALSE
)
```

Arguments

coords_df	Data frame or matrix with columns x_centroid and y_centroid (e.g. the output of SpatialExperiment::spatialCoords()). Row names must be cell IDs matching the column names of counts_matrix.
counts_matrix	Gene-by-cell count matrix (dense or sparse). Row names must be gene symbols; column names must be cell IDs present in coords_df.
bin_size	Numeric. Approximate width of each hexagonal bin in coordinate units (e.g. microns). Analogous to grid.length.x in sciderHex::gridDensity. Default 50.
min_cells	Integer. Bins containing fewer than min_cells cells are dropped from the output. Default 1.
min_total_counts	Numeric. Bins whose total counts (summed over all genes) fall below this threshold are dropped from the output, alongside the min_cells filter. Set to 0 to disable. Default 10.
group	Factor or character vector of length ncol(counts_matrix) giving the cell-type label of each cell. When supplied, a named list of per-cell-type gene-by-bin matrices is included in the output as counts_by_group. Default NULL (not computed).
verbose	Logical. If TRUE, print progress messages. Default FALSE (silent).

Value

A list with:

counts_matrix Gene-by-bin sparse count matrix (all cells combined). Column *i* corresponds to row *i* of bins.

bins An sf object of hexagonal bin polygons with an n_cells column recording how many cells fall in each bin and a total_counts column recording the summed counts per bin. Row order matches the columns of counts_matrix.

counts_by_group (Only when group is supplied.) A named list of gene-by-bin sparse matrices, one per cell-type level, with the same bin order as counts_matrix.

Examples

```
## Not run:
set.seed(42)
n <- 500
coords <- data.frame(
  x_centroid = runif(n, 0, 1000),
  y_centroid = runif(n, 0, 1000),
  row.names  = paste0("cell_", seq_len(n))
)
counts <- Matrix::Matrix(
  matrix(rpois(20L * n, lambda = 5), nrow = 20L, ncol = n,
         dimnames = list(paste0("gene_", 1:20), paste0("cell_", seq_len(n)))),
  sparse = TRUE
)
group  <- sample(c("TypeA", "TypeB"), n, replace = TRUE)
binned <- hexBinCells(coords, counts, bin_size = 100, group = group)
str(binned, max.level = 1)

## End(Not run)
```

---

is.blisa *Test if an object is a blisa object*

Description

Test if an object is a blisa object

Usage

```
is.blisa(x)
```

Arguments

x	Any R object.

Value

Logical.

Examples

```
is.blisa(list())           # FALSE
is.blisa("not a blisa")    # FALSE
```

plotCCI *Heatmap of CCI scores across all ligand-receptor pairs*

Description

Generic function. Draws a clustered heatmap (via `ComplexHeatmap`) with rows as Sender → Receiver cell-type pairs and columns as LR pairs. Row annotations colour-code the sender and receiver cell types.

Usage

```
plotCCI(x, ...)

## S3 method for class 'blisa'
plotCCI(
  x,
  top_lr = 20,
  top_pairs = 30,
  lr_pairs = NULL,
  sender = NULL,
  receiver = NULL,
  colors = NULL,
  colours = NULL,
  main = NULL,
  ...
)

## Default S3 method:
plotCCI(
  x,
  top_lr = 20,
  top_pairs = 30,
  lr_pairs = NULL,
  sender = NULL,
  receiver = NULL,
  colors = NULL,
  colours = NULL,
  main = NULL,
  ...
)
```

Arguments

`x`	A `blisa` object or a CCI scores data frame (the `CCI_scores` slot of a `blisa` object). The data frame must contain columns `Sender`, `Receiver`, and one column per LR pair.
`...`	Additional arguments passed to the relevant method.

top_lr	Integer or NULL. Number of top-ranked LR pairs (by sig_numbers) to display as columns. LR pairs in CCI_scores are already ordered by rank, so this simply takes the first top_lr columns. Ignored when lr_pairs is supplied. Default 20.
top_pairs	Integer or NULL. Number of top sender-receiver pairs to display as rows, ranked by their maximum interaction score across the displayed LR pairs (after top_lr / lr_pairs is applied). When NULL all rows are shown. Default 30.
lr_pairs	Character vector or NULL. When supplied, only these LR pair column names are shown, in the order given, overriding top_lr. Names not found in CCI_scores are dropped with a warning. Default NULL (use top_lr).
sender	Character vector or NULL. If provided, only rows where Sender is in this vector are kept. Applied independently of receiver (AND logic when both are supplied). Default NULL (all senders).
receiver	Character vector or NULL. If provided, only rows where Receiver is in this vector are kept. Applied independently of sender (AND logic when both are supplied). Default NULL (all receivers).
colors	Named character vector mapping cell-type names to colours, used for the sender/receiver row annotations. When NULL (default), colours are assigned automatically from the package palette. The British spelling colours is accepted as an alias.
colours	Alias for colors (British spelling). Ignored when colors is supplied.
main	Character or NULL. Title drawn above the heatmap (mapped to the column_title of ComplexHeatmap::Heatmap). Default NULL (no title).

Value

Invisibly returns the Heatmap object.

Methods (by class)

- plotCCI(blisa): Method for a blisa object. Extracts CCI_scores and delegates to plotCCI.default. Stops with an informative error if CCI_scores is NULL.
- plotCCI(default): Method for a CCI scores data frame (e.g. the CCI_scores slot of a blisa object).

See Also

plotCCILR for a sender-by-receiver heatmap of a single LR pair; plotCCIsummary for an aggregated sender-by-receiver heatmap across all LR pairs.

Examples

```
## Not run:
# Continuing from the blisa() example:
# result <- blisa(spe, bin_size = 50, group = "cell_type")
plotCCI(result, top_lr = 20, top_pairs = 30)

## End(Not run)
```

plotCCILR *Sender-by-receiver heatmap of CCI scores for one ligand-receptor pair*

Description

Generic function. Reshapes the CCI data frame into a receiver-by-sender cell-type matrix for one selected LR pair and draws a clustered heatmap.

Usage

```
plotCCILR(x, ...)

## S3 method for class 'blisa'
plotCCILR(x, index = 1, ligand = NULL, receptor = NULL, main = NULL, ...)

## Default S3 method:
plotCCILR(x, lr_pair, main = NULL, ...)
```

Arguments

x	A blisa object or a CCI scores data frame (the CCI_scores slot of a blisa object).
...	Additional arguments passed to the relevant method.
index	Integer. Row index into LR_results selecting the ligand-receptor pair to visualise. Ignored when both ligand and receptor are supplied. Default 1 (top-ranked pair).
ligand	Character. Ligand gene symbol. When both ligand and receptor are provided the matching LR pair is located automatically and index is ignored. Must be supplied together with receptor.
receptor	Character. Receptor gene symbol. Must be supplied together with ligand.
main	Character or NULL. Title drawn above the heatmap. When supplied, the heatmap is drawn with this overall title (via ComplexHeatmap::draw); the Heatmap object is returned invisibly. Default NULL (no title; object returned for the caller to print).
lr_pair	Character. Column name in the CCI scores data frame corresponding to the ligand-receptor pair to visualise (e.g. "CXCL12_CXCR4").

Value

A Heatmap object.

Methods (by class)

- `plotCCILR(blisa)`: Method for a `blisa` object. The LR pair is selected by `index` (default 1, the top-ranked pair) unless both `ligand` and `receptor` are supplied, in which case the matching row is located automatically and `index` is ignored. Stops with an informative error if `CCI_scores` is `NULL` or the selected LR pair has no significant hotspots.
- `plotCCILR(default)`: Method for a CCI scores data frame (e.g. the `CCI_scores` slot of a `blisa` object). The LR pair is selected by column name via `lr_pair`.

See Also

`plotCCI` for an overview heatmap across all LR pairs; `plotCCIsummary` for an aggregated sender-by-receiver heatmap.

Examples

```
## Not run:
# Continuing from the blisa() example:
# result <- blisa(spe, bin_size = 50, group = "cell_type")
plotCCILR(result, index = 1)

## End(Not run)
```

---

plotCCIspatial *Spatial map of dominant sender-receiver cell-type pairs at BLISA hotspots*

Description

For a selected ligand-receptor pair, identifies the dominant interacting cell-type pair at each significant hotspot bin and draws a spatial map of the tissue coloured by those pairs. Receiver cells are those inside hotspot bins; sender cells are drawn from the immediate neighbourhood.

Usage

```
plotCCIspatial(
  x,
  counts_by_group,
  index = 1,
  ligand = NULL,
  receptor = NULL,
  top_pairs = 30
)
```

Arguments

x	A blisa object as returned by blisa.
counts_by_group	Named list of gene-by-bin count matrices, one per cell type. Typically the counts_by_group element returned by hexBinCells. Names must match the cell-type levels.
index	Integer. Row index into x$LR_results selecting the ligand-receptor pair to visualise. Ignored when both ligand and receptor are supplied. Default 1 (top-ranked pair).
ligand	Character. Ligand gene symbol. When both ligand and receptor are provided the matching LR pair is located automatically and index is ignored. Must be supplied together with receptor.
receptor	Character. Receptor gene symbol. Must be supplied together with ligand.
top_pairs	Integer. Maximum number of distinct cell-type pairs to show in the legend; remaining pairs are grouped as "rare pairs". Default 30.

Value

A ggplot object.

See Also

plotHotspots for a significance-based spatial map of hotspot bins.

Examples

```
## Not run:
# Continuing from the blisa() example:
# result <- blisa(spe, bin_size = 50, group = "cell_type")
binned <- hexBinCells(
  as.data.frame(SpatialExperiment::spatialCoords(spe)),
  SummarizedExperiment::assay(spe, "counts"),
  bin_size = 50, group = spe$cell_type
)
plotCCIspatial(result, binned$counts_by_group, index = 1)

## End(Not run)
```

---

plotCCIsummary *Sender-by-receiver heatmap of aggregated CCI scores across LR pairs*

Description

Generic function. Aggregates CCI scores across all (or the top-ranked) ligand-receptor pairs and draws a clustered receiver-by-sender heatmap, one cell per Sender → Receiver combination.

Usage

```
plotCCIsummary(x, ...)

## S3 method for class 'blisa'
plotCCIsummary(
  x,
  top_lr = NULL,
  sender = NULL,
  receiver = NULL,
  agg_fun = sum,
  main = NULL,
  ...
)

## Default S3 method:
plotCCIsummary(
  x,
  top_lr = NULL,
  sender = NULL,
  receiver = NULL,
  agg_fun = sum,
  main = NULL,
  ...
)
```

Arguments

x	A blisa object or a CCI scores data frame (the CCI_scores slot of a blisa object).
...	Additional arguments passed to the relevant method.
top_lr	Integer or NULL. Number of top-ranked LR pairs (by sig_numbers) to include before aggregating. LR pairs in CCI_scores are already ordered by rank, so this takes the first top_lr columns. NULL (default) uses all pairs.
sender	Character vector or NULL. If provided, only rows where Sender is in this vector are kept (AND logic with receiver). Default NULL (all senders).
receiver	Character vector or NULL. If provided, only rows where Receiver is in this vector are kept (AND logic with sender). Default NULL (all receivers).
agg_fun	Function used to aggregate scores across LR pairs for each Sender → Receiver combination. Receives a numeric vector with NAs already removed. Default sum.
main	Character or NULL. Title drawn above the heatmap. When supplied, the heatmap is drawn with this overall title (via ComplexHeatmap::draw); the Heatmap object is returned invisibly. Default NULL (no title; object returned for the caller to print).

Value

A Heatmap object.

Methods (by class)

- `plotCCIsummary(blisa)`: Method for a `blisa` object. Stops with an informative error if `CCI_scores` is `NULL`.
- `plotCCIsummary(default)`: Method for a CCI scores data frame (e.g. the `CCI_scores` slot of a `blisa` object).

See Also

`plotCCILR` for a per-LR-pair version of this plot; `plotCCI` for a heatmap with LR pairs as columns.

Examples

```
## Not run:
# Continuing from the blisa() example:
# result <- blisa(spe, bin_size = 50, group = "cell_type")
plotCCIsummary(result)
plotCCIsummary(result, top_lr = 10, agg_fun = mean)

## End(Not run)
```

---

plotHotspots *Spatial hotspot map for one ligand-receptor pair*

Description

Generic function. Plots each bin coloured by significance status: empty, non-significant, or significant hotspot (continuous gradient of -log10 p-value or 1 - p-value).

Usage

```
plotHotspots(x, ...)

## S3 method for class 'blisa'
plotHotspots(
  x,
  index = 1,
  ligand = NULL,
  receptor = NULL,
  log_pval = TRUE,
  p_cutoff = NULL,
  ...
)
```

Arguments

x	A blisa object.
...	Additional arguments passed to the method.
index	Integer. Row index into LR_results selecting the ligand-receptor pair to visualise. Ignored when both ligand and receptor are supplied. Default 1 (top-ranked pair).
ligand	Character. Ligand gene symbol. When both ligand and receptor are provided the matching LR pair is located automatically and index is ignored. Must be supplied together with receptor.
receptor	Character. Receptor gene symbol. Must be supplied together with ligand.
log_pval	Logical. If TRUE (default), colour significant bins by -log10(p-value). If FALSE, use 1 - p-value.
p_cutoff	Numeric or NULL. When NULL (default), the pre-computed hotspot bins stored in the blisa object are used, reflecting the p_cutoff and High-High quadrant classification applied during blisa. When a numeric value is supplied, bins are re-defined on the fly as those with all_pval <= p_cutoff and quadrant label "High-High" (from the stored all_quadrant), giving an exact re-threshold consistent with the original classification.

Value

A ggplot object.

Methods (by class)

- plotHotspots(blisa): Method for a blisa object.

Examples

```
## Not run:
# Continuing from the blisa() example:
# result <- blisa(spe, bin_size = 50, group = "cell_type")
plotHotspots(result, index = 1)
plotHotspots(result, index = 1, log_pval = FALSE)

## End(Not run)
```

---

plotLRrank *Dot plot ranking LR pairs by number of significant hotspot bins*

Description

Generic function for ranking LR pairs. Dispatches on the class of x:

- plotLRrank.blisa accepts a blisa object and uses its LR_results slot directly.
- plotLRrank.data.frame accepts the LR_results data frame directly.

Usage

```
plotLRrank(x, ...)

## S3 method for class 'blisa'
plotLRrank(x, top = 30, pt_size = 4, flip = FALSE, ...)

## S3 method for class 'data.frame'
plotLRrank(x, top = 30, pt_size = 4, flip = FALSE, ...)
```

Arguments

x	A blisa object or a data frame of LR results. The data frame must contain columns sig_numbers and annotation.
...	Additional arguments passed to the relevant method.
top	Integer or NULL. Number of top LR pairs (by sig_numbers) to display. Default 30.
pt_size	Numeric. Point size passed to geom_point. Default 4.
flip	Logical. When TRUE, LR pairs are placed on the x-axis and the hotspot count on the y-axis (vertical orientation). Default FALSE (LR pairs on y-axis, horizontal orientation).

Value

A ggplot object.

Methods (by class)

- plotLRrank(blisa): Method for a blisa object. Extracts LR_results and delegates to plotLRrank.data.frame.
- plotLRrank(data.frame): Method for a data frame of LR results (e.g. the LR_results slot of a blisa object).

Examples

```
## Not run:
# Continuing from the blisa() example:
# result <- blisa(spe, bin_size = 50, group = "cell_type")
plotLRrank(result, top = 30)
plotLRrank(result, top = 20, flip = TRUE)

## End(Not run)
```

runCCI *Score cell-cell interactions from BLISA hotspots*

Description

Generic function for scoring cell-cell interactions. Dispatches on the class of `x`:

- `runCCI.blisa` accepts a `blisa` object. If `CCI_scores` are already present and `overwrite = FALSE` (the default), the object is returned unchanged. Set `overwrite = TRUE` with a `counts_by_group` to recompute and replace existing scores. If no scores exist, `counts_by_group` must be supplied and scores are computed and attached.
- `runCCI.default` performs the raw computation given a `blisa` object and a `counts_by_group` list, returning only the scores data frame. Used internally by `runCCI.blisa` and `blisa.default`.

Usage

```
runCCI(x, ...)

## S3 method for class 'blisa'
runCCI(x, counts_by_group = NULL, overwrite = FALSE, ...)

## Default S3 method:
runCCI(x, counts_by_group, ...)
```

Arguments

`x`	A `blisa` object.
`...`	Additional arguments passed to the relevant method.
`counts_by_group`	Named list of gene-by-bin sparse count matrices, one per group level (e.g. cell type). Typically the `counts_by_group` element of the list returned by `hexBinCells` when `group` is supplied. Names must match the group levels. Required when `x$CCI_scores` is `NULL` or when `overwrite = TRUE`.
`overwrite`	Logical. If `FALSE` (default) and `x$CCI_scores` is already populated, the object is returned unchanged. If `TRUE` and `counts_by_group` is supplied, existing scores are recomputed and replaced.

Value

See individual method documentation.

`runCCI.blisa`: the input `blisa` object with `CCI_scores` populated (a wide data frame – rows are `"Sender->Receiver"` group pairs, columns are LR pairs).

`runCCI.default`: a data frame with `"Sender->Receiver"` row names and one column per significant LR pair containing the interaction score `0.5 * log2(receiver * sender + 1)`.

Methods (by class)

- runCCI(blisa): Method for a blisa object. If CCI_scores are already present and overwrite = FALSE (the default), the object is returned unchanged. Set overwrite = TRUE with a counts_by_group to recompute and replace existing scores. If no scores exist, counts_by_group must be supplied and scores are computed and attached to x$CCI_scores.
- runCCI(default): Default method. Performs the raw CCI computation and returns only the scores data frame. Typically called internally; use runCCI.blisa to compute and attach scores to a blisa object in one step.

Examples

```
## Not run:
# Continuing from the blisa() example:
# result <- blisa(spe, bin_size = 50, group = "cell_type")

# CCI is computed automatically when group is supplied to blisa();
# access the scores directly:
head(result$CCI_scores)

# Or compute / recompute scores explicitly:
binned <- hexBinCells(
  as.data.frame(SpatialExperiment::spatialCoords(spe)),
  SummarizedExperiment::assay(spe, "counts"),
  bin_size = 50, group = spe$cell_type
)
result2 <- runCCI(result, counts_by_group = binned$counts_by_group,
                  overwrite = TRUE)

## End(Not run)
```

Index