Package: COMMOTR (via r-universe)

May 26, 2026

Type Package

Title Screening Cell-Cell Communication in Spatial Transcriptomics via Collective Optimal Transport

Version 1.0.0

Date 2026-01-25

Description Infers cell-cell communication in spatial transcriptomics data using collective optimal transport. The method models ligand-receptor interactions as optimal transport problems with spatial distance constraints, providing communication direction inference, cluster-level analysis, and comprehensive visualization. This is an R implementation based on the COMMOT algorithm (Cang et al., Nature Methods, 2023).

License MIT + file LICENSE

URL https://zaoqu-liu.github.io/COMMOTR/, https://github.com/Zaoqu-Liu/COMMOTR

BugReports https://github.com/Zaoqu-Liu/COMMOTR/issues

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Depends R (>= 4.0.0)

Imports Rcpp (>= 1.0.0), Matrix, methods, stats, Seurat (>= 4.0.0), SeuratObject, future, future.apply, ggplot2, igraph, dplyr, tidyr, scales, grDevices, rlang, RColorBrewer

LinkingTo Rcpp, RcppArmadillo

Suggests circlize, testthat (>= 3.0.0), knitr, rmarkdown, ggraph, tidygraph, viridis, patchwork

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation yes

SystemRequirements C++11

Config/pak/sysreqs cmake libglpk-dev make libicu-dev libpng-dev libuv1-dev libxml2-dev libssl-dev python3 zlib1g-dev

Repository https://zaoqu-liu.r-universe.dev

Date/Publication 2026-01-24 19:36:13 UTC

RemoteUrl https://github.com/Zaoqu-Liu/COMMOTR

RemoteRef main

RemoteSha d201e5449329a215401be3f9d9b64410bfe5e669

Contents

`COMMOTR-package` *COMMOTR: Cell-Cell Communication via Collective Optimal Transport*

Description

Infer cell-cell communication in spatial transcriptomics data using collective optimal transport (COT). COMMOTR provides a complete workflow for analyzing ligand-receptor interactions with spatial constraints.

Details

COMMOTR implements the COMMOT algorithm from Cang et al. (Nature Methods, 2023) in R with high-performance C++ backends for optimal transport computations.

Core Analysis Functions:

- `spatial_communication`: Infer cell-cell communication
- `communication_direction`: Compute communication directions
- `cluster_communication`: Cluster-level analysis

Data Functions:

- `ligand_receptor_database`: Load LR databases
- `filter_lr_database`: Filter LR pairs by expression

Visualization Functions:

- `plot_cell_communication`: Cell-level plots with vector fields
- `plot_cluster_communication`: Cluster-level network plots
- `plot_communication_dotplot`: Dotplot visualization

Author(s)

Zaoqu Liu <liuzaoqu@163.com>

References

Cang, Z., Zhao, Y., Almet, A.A. et al. Screening cell-cell communication in spatial transcriptomics via collective optimal transport. Nature Methods 20, 218-228 (2023). doi:10.1038/s41592022-017284

See Also

- GitHub: https://github.com/Zaoqu-Liu/COMMOTR
- Original COMMOT: https://github.com/zcang/COMMOT

`cluster_communication` *Cluster-Level Communication Analysis*

Description

Aggregate and analyze cell-cell communication at cluster level.

Aggregate cell-cell communication to cluster-cluster level and optionally perform permutation testing.

Usage

```
cluster_communication(
  seurat_obj,
  database_name,
  clustering,
  lr_pair = NULL,
  pathway_name = NULL,
  n_permutations = 100L,
  seed = NULL,
  verbose = TRUE
)
```

Arguments

`seurat_obj`	Seurat object with COMMOTR results.
`database_name`	Name of database.
`clustering`	Name of metadata column containing cluster assignments, or a factor/character vector of cluster assignments.
`lr_pair`	Specific LR pair (if NULL, uses pathway_name or total).
`pathway_name`	Pathway name (if lr_pair is NULL).
`n_permutations`	Number of permutations for significance testing (default: 100). Set to 0 to skip.
`seed`	Random seed for permutation.
`verbose`	Print progress messages.

Details

For each cluster pair (i, j), computes the total communication from cells in cluster i to cells in cluster j. If permutation testing is enabled, p-values are computed by randomly permuting cluster labels.

Value

Seurat object with cluster communication results stored.

Examples

```
## Not run:
# After running spatial_communication
seurat_obj <- cluster_communication(
    seurat_obj,
    database_name = "CellChat",
    clustering = "seurat_clusters",
    pathway_name = "TGFb",
    n_permutations = 100
)

# Plot results
plot_cluster_communication(
    seurat_obj,
    database_name = "CellChat",
    clustering = "seurat_clusters",
    key = "TGFb"
)

## End(Not run)
```

`cluster_communication_spatial_permutation` *Cluster communication with spatial permutation*

Description

Permutation test that preserves spatial structure by rotating coordinates.

Usage

```
cluster_communication_spatial_permutation(
  seurat_obj,
  database_name,
  clustering,
  lr_pair = NULL,
  pathway_name = NULL,
  spatial_coords = NULL,
  n_permutations = 100L,
  random_seed = NULL,
  verbose = TRUE
)
```

Arguments

`seurat_obj`	Seurat object with COMMOTR results.
`database_name`	Database name.

clustering	Clustering column or vector.
lr_pair	LR pair name.
pathway_name	Pathway name.
spatial_coords	Spatial coordinates.
n_permutations	Number of permutations.
random_seed	Random seed.
verbose	Print progress.

Value

Seurat object with results.

`cluster_position` *Compute cluster centroid positions*

Description

Compute cluster centroid positions

Usage

```
cluster_position(seurat_obj, clustering, spatial_coords = NULL)
```

Arguments

seurat_obj	Seurat object.
clustering	Clustering column or vector.
spatial_coords	Spatial coordinates.

Value

Data frame with cluster centroids.

`communication_deg_clustering` *Cluster communication-dependent genes*

Description

Group DEGs based on their expression patterns using Leiden clustering.

Usage

```
communication_deg_clustering(
  df_deg,
  df_yhat,
  deg_clustering_npc = 10L,
  deg_clustering_knn = 5L,
  deg_clustering_res = 1,
  n_deg_genes = 200L,
  p_value_cutoff = 0.05
)
```

Arguments

`df_deg`	DEG results from communication_deg_detection().
`df_yhat`	Smoothed expression from communication_deg_detection().
`deg_clustering_npc`	Number of PCs for embedding.
`deg_clustering_knn`	Number of neighbors for KNN graph.
`deg_clustering_res`	Leiden resolution parameter.
`n_deg_genes`	Number of top DEGs to cluster.
`p_value_cutoff`	P-value cutoff for including genes.

Value

List with:

- df_metadata: Gene metadata with cluster assignments
- df_yhat: Smoothed expression for clustered genes

communication_deg_detection *Detect communication-dependent genes*

Description

Identify genes whose expression correlates with communication intensity. Uses tradeSeq for GAM-based testing when available.

Usage

```
communication_deg_detection(
  seurat_obj,
  database_name,
  pathway_name = NULL,
  lr_pair = c("total", "total"),
  summary = c("sender", "receiver"),
  assay = NULL,
  n_var_genes = 2000L,
  var_genes = NULL,
  nknots = 6L,
  n_deg_genes = NULL,
  n_points = 50L,
  deg_pvalue_cutoff = 0.05,
  use_tradeseq = TRUE,
  verbose = TRUE
)
```

Arguments

seurat_obj	Seurat object with COMMOTR results.
database_name	Database name.
pathway_name	Signaling pathway name (optional).
lr_pair	LR pair as c("ligand", "receptor") (optional).
summary	"sender" or "receiver" signal to use.
assay	Assay to use for count data.
n_var_genes	Number of most variable genes to test.
var_genes	Specific genes to test (overrides n_var_genes).
nknots	Number of spline knots for GAM.
n_deg_genes	Number of top genes for expression pattern.
n_points	Number of points for smoothed pattern.
deg_pvalue_cutoff	P-value cutoff for significant genes.
use_tradeseq	Use tradeSeq package if available (default: TRUE).
verbose	Print progress.

Details

When tradeSeq is available, uses GAM-based differential expression testing. Otherwise, uses a simpler F-test based approach.

Value

List with:

- df_deg: Data frame with Wald statistics, df, and p-values
- df_yhat: Data frame with smoothed expression patterns

References

Van den Berge, K., et al. (2020). Trajectory-based differential expression analysis for single-cell sequencing data. Nature Communications, 11(1), 1-13.

`communication_direction`

Communication Direction Analysis

Description

Infer spatial vector fields describing communication directions.

Compute spatial vector fields representing the average direction of cell-cell communication signals from senders to receivers.

Usage

```
communication_direction(
  seurat_obj,
  database_name,
  lr_pair = NULL,
  pathway_name = NULL,
  spatial_coords = NULL,
  k = 5L,
  bandwidth = NULL,
  kernel = c("exp", "lorentz"),
  normalize = TRUE,
  verbose = TRUE
)
```

Arguments

seurat_obj	Seurat object with COMMOTR results from spatial_communication.
database_name	Name of database used in spatial_communication.
lr_pair	Specific LR pair name (e.g., "Tgfb1-Tgfbr1_Tgfbr2"). If NULL, uses pathway_name.
pathway_name	Pathway name. Used if lr_pair is NULL. If both NULL, uses "total".
spatial_coords	Spatial coordinates: NULL (auto-detect), character (reduction name), or matrix.
k	Number of nearest neighbors for spatial smoothing (default: 5).
bandwidth	Bandwidth for kernel smoothing. If NULL, estimated from k.
kernel	Smoothing kernel: "exp" (exponential, default) or "lorentz".
normalize	Logical, normalize vectors to unit length (default: TRUE).
verbose	Logical, print progress messages (default: TRUE).

Details

For each cell, the function computes:

- Sender vector field: weighted average direction toward receiver cells
- Receiver vector field: weighted average direction from sender cells

The weights are communication matrix values smoothed with a spatial kernel. The resulting vector fields can be visualized using plot_cell_communication.

Value

Seurat object with vector field results stored in the COMMOTR results structure.

See Also

spatial_communication, plot_cell_communication, communication_spatial_autocorrelation

Examples

```
## Not run:
# Compute direction for TGFb pathway
seurat_obj <- communication_direction(
    seurat_obj,
    database_name = "CellChat",
    pathway_name = "TGFb",
    k = 5
)

# Visualize
plot_cell_communication(seurat_obj, "CellChat", pathway_name = "TGFb")

## End(Not run)
```

`communication_impact` *Analyze communication impact on gene expression*

Description

Compute the impact of communication on target genes.

Usage

```
communication_impact(
  seurat_obj,
  database_name,
  pathway_name = NULL,
  pathway_sum_only = FALSE,
  heteromeric_delimiter = "_",
  normalize = FALSE,
  method = c("partial_corr", "semipartial_corr", "treebased_score"),
  corr_method = c("spearman", "pearson"),
  tree_method = c("rf", "gbt"),
  tree_ntrees = 100L,
  tree_repeat = 100L,
  tree_max_depth = 5L,
  tree_max_features = "sqrt",
  tree_learning_rate = 0.1,
  tree_subsample = 1,
  tree_combined = FALSE,
  ds_genes,
  bg_genes = 100L
)
```

Arguments

`seurat_obj`	Seurat object with COMMOTR results.
`database_name`	Database name.
`pathway_name`	Signaling pathway to analyze (optional).
`pathway_sum_only`	Only analyze pathway-level, not individual LR pairs.
`heteromeric_delimiter`	Delimiter for heteromeric complexes.
`normalize`	Normalize expression before analysis.
`method`	Impact analysis method: • "partial_corr": Partial correlation • "semipartial_corr": Semipartial correlation • "treebased_score": Random forest importance

corr_method	Correlation method: "spearman" or "pearson".
tree_method	Tree method: "rf" or "gbt".
tree_ntrees	Number of trees.
tree_repeat	Number of repeats.
tree_max_depth	Maximum tree depth.
tree_max_features	Max features per split.
tree_learning_rate	Learning rate (GBT).
tree_subsample	Subsample ratio.
tree_combined	Use combined model for all features.
ds_genes	Target genes to analyze.
bg_genes	Background genes (list or integer for top variable).

Value

Data frame with impact scores.

`communication_spatial_autocorrelation` *Compute spatial autocorrelation of communication direction*

Description

Compute Moran's I statistic for the vector field to assess spatial coherence of communication directions.

Compute Moran's I for communication vector fields.

Usage

```
communication_spatial_autocorrelation(
  seurat_obj,
  database_name,
  keys,
  method = "Moran",
  normalize_vf = FALSE,
  summary = c("sender", "receiver"),
  weight_bandwidth = NULL,
  weight_k = 10L,
 weight_function = c("triangular", "uniform", "quadratic", "quartic", "gaussian"),
  weight_row_standardize = FALSE,
  n_permutations = 999L
)
```

```
communication_spatial_autocorrelation(
  seurat_obj,
  database_name,
  keys,
  method = "Moran",
  normalize_vf = FALSE,
  summary = c("sender", "receiver"),
  weight_bandwidth = NULL,
  weight_k = 10L,
 weight_function = c("triangular", "uniform", "quadratic", "quartic", "gaussian"),
  weight_row_standardize = FALSE,
  n_permutations = 999L
)
```

Arguments

seurat_obj	Seurat object with direction results.
database_name	Database name.
keys	Communication keys.
method	Currently only "Moran".
normalize_vf	Normalize vectors to unit length.
summary	"sender" or "receiver".
weight_bandwidth	Bandwidth for kernel weights.
weight_k	Number of neighbors.
weight_function	Kernel function.
weight_row_standardize	Row-standardize weights.
n_permutations	Number of permutations for p-value.
key	LR pair or pathway name.
field	"sender" or "receiver" vector field.
spatial_coords	Spatial coordinates.
k	Number of neighbors for spatial weights.
seed	Random seed for reproducibility.

Details

Moran's I measures spatial autocorrelation. Positive values indicate that nearby cells have similar communication directions (spatially coherent signaling), while values near zero indicate random directions.

Value

List with Moran's I statistic, expected value, and p-value.

Data frame with Moran's I and p-values for each key.

database_info *Built-in Ligand-Receptor Database Information*

Description

COMMOTR includes curated ligand-receptor databases from CellChat and CellPhoneDB. These databases are stored as RDS files and loaded via `ligand_receptor_database`.

Details

Available databases:

- **CellChat (Mouse)**: 2,019 ligand-receptor interactions
- **CellChat (Human)**: 1,939 ligand-receptor interactions
- **CellChat (Zebrafish)**: 2,774 ligand-receptor interactions
- **CellPhoneDB v4.0 (Mouse)**: 1,410 ligand-receptor interactions
- **CellPhoneDB v4.0 (Human)**: 1,680 ligand-receptor interactions

Each database contains:

ligand Ligand gene name(s), underscore-separated for complexes

receptor Receptor gene name(s), underscore-separated for complexes

pathway Signaling pathway name

signaling_type Type: Secreted Signaling, Cell-Cell Contact, or ECM-Receptor

References

CellChat: Jin, S. et al. Inference and analysis of cell-cell communication using CellChat. Nat Commun 12, 1088 (2021).

CellPhoneDB: Efremova, M. et al. CellPhoneDB: inferring cell-cell communication from combined expression of multi-subunit ligand-receptor complexes. Nat Protoc 15, 1484-1506 (2020).

See Also

`ligand_receptor_database`, `filter_lr_database`

downstream_analysis *Downstream Analysis Functions*

Description

Downstream analysis for cell-cell communication results, following Cang et al., Nature Methods 2023.

`filter_lr_database` *Filter ligand-receptor database by expression*

Description

Filter LR pairs to keep only those with sufficient expression in the dataset.

Usage

```
filter_lr_database(
  df_ligrec,
  seurat_obj,
  assay = NULL,
  slot = "data",
  heteromeric = TRUE,
  heteromeric_delimiter = "_",
  heteromeric_rule = c("min", "ave"),
  min_cell = 100L,
  min_cell_pct = 0.05,
  verbose = TRUE
)
```

Arguments

`df_ligrec`	Data frame from `ligand_receptor_database`.
`seurat_obj`	Seurat object with expression data.
`assay`	Assay to use (default: DefaultAssay).
`slot`	Expression slot/layer (default: "data").
`heteromeric`	Whether ligands/receptors can be multi-subunit complexes (default: TRUE).
`heteromeric_delimiter`	Character separating subunit genes (default: "_").
`heteromeric_rule`	Aggregation rule for complex expression: "min" (default) uses minimum subunit, "ave" uses average.
`min_cell`	Minimum number of cells expressing (alternative criterion).
`min_cell_pct`	Minimum percentage of cells expressing (default: 0.05).
`verbose`	Print progress messages (default: TRUE).

Value

Filtered data frame of ligand-receptor pairs.

Examples

```
## Not run:
df_lr <- ligand_receptor_database("CellChat", "mouse")
df_lr_filtered <- filter_lr_database(df_lr, seurat_obj, min_cell_pct = 0.05)

## End(Not run)
```

`get_communication_matrix` *Get communication matrix from results*

Description

Get communication matrix from results

Usage

```
get_communication_matrix(
  obj,
  database_name,
  lr_pair = NULL,
  pathway_name = NULL
)
```

Arguments

`obj`	Seurat object
`database_name`	Database name
`lr_pair`	LR pair name (e.g., "Tgfb1-Tgfbr1_Tgfbr2")
`pathway_name`	Pathway name (alternative to lr_pair)

Value

Sparse matrix or NULL

get_communication_results

Get COMMOTR results from Seurat object

Description

Get COMMOTR results from Seurat object

Usage

```
get_communication_results(obj, database_name)
```

Arguments

obj	Seurat object
database_name	Database name used in spatial_communication()

Value

Results list or NULL if not found

get_sender_receiver_df

Get sender/receiver summary data frame

Description

Get sender/receiver summary data frame

Usage

```
get_sender_receiver_df(obj, database_name, type = "sender")
```

Arguments

obj	Seurat object
database_name	Database name
type	"sender" or "receiver"

Value

Data frame

group_cell_communication *Group cell-level communication patterns*

Description

Identify groups of similar cell-cell communication patterns using graph embedding.

Usage

```
group_cell_communication(
  seurat_obj,
  database_name,
  keys,
  bin_method = c("gaussian_mixture", "kmeans"),
  bin_append_zeros = c("full", "match"),
  bin_random_state = 1L,
  bin_cutoff = 0,
  knn = 2L,
  dissimilarity_method = c("spectral", "heat_kernel"),
  leiden_k = 5L,
  leiden_resolution = 1,
  leiden_random_seed = 1L,
  leiden_n_iterations = -1L
)
```

Arguments

seurat_obj	Seurat object with COMMOTR results.
database_name	Database name.
keys	Communication keys to analyze.
bin_method	Binarization method: "gaussian_mixture" or "kmeans".
bin_append_zeros	How to handle zeros: "full" or "match".
bin_random_state	Random seed for binarization.
bin_cutoff	Force small values to zero.
knn	KNN for spatial graph.
dissimilarity_method	Embedding method: "spectral" or "heat_kernel".
leiden_k	KNN for Leiden clustering.
leiden_resolution	Resolution parameter.

leiden_random_seed
: Random seed.

leiden_n_iterations
: Max iterations.

Value

List with cluster assignments and dissimilarity matrix.

`group_cluster_communication` *Group cluster communications*

Description

Group similar cluster-cluster communication patterns based on graph structure similarity.

Identify groups of similar cluster-cluster communication patterns.

Usage

```
group_cluster_communication(
  seurat_obj,
  clustering,
  cluster_permutation_type = c("label", "spatial"),
  keys,
  p_value_cutoff = 0.05,
  quantile_cutoff = 0.99,
  dissimilarity_method = c("jaccard", "jaccard_weighted", "global_structure"),
  leiden_k = 5L,
  leiden_resolution = 1,
  leiden_random_seed = 1L,
  leiden_n_iterations = -1L,
  d_global_structure_weights = c(0.45, 0.45, 0.1)
)

group_cluster_communication(
  seurat_obj,
  clustering,
  cluster_permutation_type = c("label", "spatial"),
  keys,
  p_value_cutoff = 0.05,
  quantile_cutoff = 0.99,
  dissimilarity_method = c("jaccard", "jaccard_weighted", "global_structure"),
  leiden_k = 5L,
  leiden_resolution = 1,
  leiden_random_seed = 1L,
  leiden_n_iterations = -1L,
```

```
  d_global_structure_weights = c(0.45, 0.45, 0.1)
)
```

Arguments

seurat_obj	Seurat object with cluster communication results.
clustering	Name of clustering.
cluster_permutation_type	"label" or "spatial".
keys	Communication keys to analyze.
p_value_cutoff	P-value cutoff for edges.
quantile_cutoff	Quantile cutoff for edges.
dissimilarity_method	Method for comparing graphs: "jaccard", "jaccard_weighted", or "global_structure".
leiden_k	KNN parameter for Leiden.
leiden_resolution	Resolution for Leiden.
leiden_random_seed	Random seed.
leiden_n_iterations	Max iterations.
d_global_structure_weights	Weights for global structure method.
database_name	Database name.
n_groups	Number of groups to create (default: auto).
method	Grouping method: "hierarchical" (default), "kmeans".
distance_metric	Distance between communication graphs: "jaccard" (default), "correlation", "euclidean".
verbose	Print progress.

Value

Data frame with grouping assignments and distances.

List with cluster assignments and dissimilarity matrix.

`group_communication_direction` *Group communication direction patterns*

Description

Cluster cells based on similarity of their communication direction vectors.

Identify groups of similar communication direction vector fields.

Usage

```
group_communication_direction(
  seurat_obj,
  database_name,
  keys,
  summary = c("sender", "receiver"),
  knn_smoothing = -1L,
  normalize_vf = c("quantile", "unit_norm", NULL),
  normalize_quantile = 0.99,
  dissimilarity_method = "dot_product",
  leiden_k = 5L,
  leiden_resolution = 1,
  leiden_random_seed = 1L,
  leiden_n_iterations = -1L
)

group_communication_direction(
  seurat_obj,
  database_name,
  keys,
  summary = c("sender", "receiver"),
  knn_smoothing = -1L,
  normalize_vf = c("quantile", "unit_norm", NULL),
  normalize_quantile = 0.99,
  dissimilarity_method = "dot_product",
  leiden_k = 5L,
  leiden_resolution = 1,
  leiden_random_seed = 1L,
  leiden_n_iterations = -1L
)
```

Arguments

`seurat_obj`	Seurat object with direction results.
`database_name`	Database name.
`keys`	Communication keys.

summary	"sender" or "receiver".
knn_smoothing	KNN for smoothing (-1 for none).
normalize_vf	Normalization: "quantile", "unit_norm", or NULL.
normalize_quantile	Quantile for normalization.
dissimilarity_method	Currently only "dot_product".
leiden_k	KNN for Leiden.
leiden_resolution	Resolution parameter.
leiden_random_seed	Random seed.
leiden_n_iterations	Max iterations.
key	LR pair or pathway name (default: "total").
field	"sender", "receiver", or "both" (default).
n_clusters	Number of clusters (default: auto).
method	Clustering method: "kmeans" (default), "hierarchical".
verbose	Print progress messages.

Value

Seurat object with cluster assignments in metadata.

List with cluster assignments and dissimilarity matrix.

---

ligand_receptor_database

Get ligand-receptor database

Description

Load built-in ligand-receptor interaction databases for cell-cell communication analysis.

Usage

```
ligand_receptor_database(
  database = c("CellChat", "CellPhoneDB"),
  species = c("mouse", "human", "zebrafish"),
  signaling_type = "Secreted Signaling"
)
```

Arguments

database	Database name: "CellChat" (default) or "CellPhoneDB".
species	Species: "mouse" (default), "human", or "zebrafish" (CellChat only).
signaling_type	Type of signaling to include: "Secreted Signaling" (default), "Cell-Cell Contact", "ECM-Receptor" (CellChat), or NULL for all.

Value

A data frame with columns:

ligand Ligand gene name(s), underscore-separated for complexes

receptor Receptor gene name(s), underscore-separated for complexes

pathway Signaling pathway name

signaling_type Type of signaling interaction

References

CellChat: Jin, S. et al. Inference and analysis of cell-cell communication using CellChat. Nat Commun 12, 1088 (2021).

CellPhoneDB: Efremova, M. et al. CellPhoneDB: inferring cell-cell communication from combined expression of multi-subunit ligand-receptor complexes. Nat Protoc 15, 1484-1506 (2020).

Examples

```
# Load mouse secreted signaling from CellChat
df_lr <- ligand_receptor_database("CellChat", "mouse", "Secreted Signaling")
head(df_lr)

# Load all human CellChat interactions
df_lr_all <- ligand_receptor_database("CellChat", "human", NULL)
```

---

lr_database *Ligand-Receptor Database Functions*

Description

Functions for loading and filtering ligand-receptor databases

plot_cell_communication *Plot cell-level communication*

Description

Visualize cell-cell communication using spatial plots with vector fields, arrows, or stream plots.

Usage

```
plot_cell_communication(
  seurat_obj,
  database_name,
  lr_pair = NULL,
  pathway_name = NULL,
  spatial_coords = NULL,
  plot_method = c("grid", "cell", "stream"),
  color_by = "signal",
  signal_type = c("sender", "receiver"),
  arrow_scale = 1,
  grid_resolution = 30L,
  point_size = 1,
  point_alpha = 0.8,
  arrow_color = "black",
  arrow_alpha = 0.6,
  color_palette = NULL,
  title = NULL,
  legend_title = NULL,
  theme_style = c("minimal", "void", "classic")
)
```

Arguments

seurat_obj	Seurat object with COMMOTR results.
database_name	Database name.
lr_pair	LR pair name (if NULL, uses pathway_name).
pathway_name	Pathway name (if lr_pair is NULL).
spatial_coords	Spatial coordinates or reduction name.
plot_method	Method: "cell" (arrows at cells), "grid" (gridded arrows), "stream" (streamlines).
color_by	Color points by: "signal", "sender", "receiver", or metadata column name.
signal_type	For vector field: "sender" or "receiver".
arrow_scale	Scale factor for arrows (default: 1).
grid_resolution	Resolution for grid method (default: 30).

point_size	Size of cell points (default: 1).
point_alpha	Transparency of points (default: 0.8).
arrow_color	Color for arrows (default: "black").
arrow_alpha	Transparency of arrows (default: 0.6).
color_palette	Color palette for continuous coloring.
title	Plot title (default: auto).
legend_title	Legend title.
theme_style	Theme: "minimal" (default), "void", "classic".

Value

A ggplot2 object.

Examples

```
## Not run:
# Plot with grid method
plot_cell_communication(
    seurat_obj,
    database_name = "CellChat",
    pathway_name = "TGFb",
    plot_method = "grid",
    color_by = "signal"
)

## End(Not run)
```

---

plot_chord_diagram *Plot chord diagram*

Description

Create a chord diagram showing cluster-cluster communication.

Usage

```
plot_chord_diagram(
  seurat_obj,
  database_name,
  clustering,
  key = "total",
  p_threshold = 0.05,
  min_weight = NULL,
  colors = NULL,
  ...
)
```

Arguments

seurat_obj	Seurat object with cluster communication results.
database_name	Database name.
clustering	Clustering name.
key	LR pair or pathway name.
p_threshold	P-value threshold.
min_weight	Minimum weight to show.
colors	Named vector of colors for clusters.
...	Additional arguments passed to circlize functions.

Value

Invisibly returns the adjacency matrix.

---

plot_cluster_communication

Plot cluster-level communication network

Description

Visualize cluster-cluster communication as a network graph.

Usage

```
plot_cluster_communication(
  seurat_obj,
  database_name,
  clustering,
  key = "total",
  layout = c("circle", "spatial", "fr"),
  edge_width_range = c(0.5, 3),
  edge_color_by = "weight",
  node_size_by = "n_cells",
  node_size_range = c(5, 15),
  show_self_loops = FALSE,
  p_threshold = 0.05,
  min_weight = NULL,
  edge_color_palette = NULL,
  node_color = NULL,
  title = NULL
)
```

Arguments

seurat_obj	Seurat object with cluster communication results.
database_name	Database name.
clustering	Clustering name.
key	LR pair or pathway name.
layout	Network layout: "spatial" (uses cluster positions), "circle", "fr" (Fruchterman-Reingold).
edge_width_range	Range for edge widths (default: c(0.5, 3)).
edge_color_by	Color edges by: "weight", "pvalue", or fixed color.
node_size_by	Size nodes by: "n_cells", "total_signal", or fixed.
node_size_range	Range for node sizes (default: c(3, 10)).
show_self_loops	Show self-communication (default: FALSE).
p_threshold	P-value threshold for showing edges (default: 0.05).
min_weight	Minimum edge weight to show.
edge_color_palette	Color palette for edges.
node_color	Color or palette for nodes.
title	Plot title.

Value

A ggplot2 object.

---

plot_communication_deg

Plot communication-dependent genes

Description

Visualize expression patterns of communication-dependent genes.

Usage

```
plot_communication_deg(
  df_deg,
  df_yhat,
  df_metadata = NULL,
  top_n = 30L,
  cluster = TRUE,
  color_palette = NULL,
  title = NULL
)
```

Arguments

df_deg	DEG results from communication_deg_detection().
df_yhat	Smoothed expression from communication_deg_detection().
df_metadata	Optional metadata from communication_deg_clustering().
top_n	Number of top genes to plot.
cluster	Show cluster annotation.
color_palette	Color palette for expression.
title	Plot title.

Value

A ggplot2 object.

---

plot_communication_dotplot

Plot communication dotplot

Description

Create a dotplot showing communication strength across multiple LR pairs or pathways.

Usage

```
plot_communication_dotplot(
  seurat_obj,
  database_name,
  clustering,
  keys = NULL,
  cluster_pairs = NULL,
  top_n_pairs = 20L,
  size_by = "communication",
  color_by = "communication",
  size_range = c(1, 6),
  p_threshold = 0.05,
  title = NULL
)
```

Arguments

seurat_obj	Seurat object with cluster communication results.
database_name	Database name.
clustering	Clustering name.
keys	LR pairs or pathways to show.

cluster_pairs	Specific cluster pairs to show (data.frame with sender and receiver columns). If NULL, shows top pairs.
top_n_pairs	Show top N cluster pairs by communication strength.
size_by	Dot size: "communication" (default) or "pvalue".
color_by	Dot color: "communication", "pvalue", or fixed.
size_range	Range for dot sizes.
p_threshold	P-value threshold.
title	Plot title.

Value

A ggplot2 object.

`plot_communication_groups` *Plot grouping results*

Description

Visualize grouped communication patterns using UMAP-like embedding.

Usage

```
plot_communication_groups(
  D,
  clusters,
  point_size = 5,
  label_points = TRUE,
  title = NULL
)
```

Arguments

D	Dissimilarity matrix.
clusters	Cluster assignments.
point_size	Point size.
label_points	Show labels.
title	Plot title.

Value

A ggplot2 object.

plot_communication_heatmap *Plot communication heatmap*

Description

Create a heatmap of cluster-cluster communication.

Usage

```
plot_communication_heatmap(
  seurat_obj,
  database_name,
  clustering,
  key = "total",
  show_pvalue = TRUE,
  color_palette = NULL,
  title = NULL
)
```

Arguments

seurat_obj	Seurat object.
database_name	Database name.
clustering	Clustering name.
key	LR pair or pathway name.
show_pvalue	Show significance stars.
color_palette	Color palette for heatmap.
title	Plot title.

Value

A ggplot2 object.

plot_communication_impact *Plot communication impact*

Description

Visualize communication impact on target genes.

Usage

```
plot_communication_impact(
  impact_df,
  top_n = 15L,
  show_sender = TRUE,
  show_receiver = TRUE,
  color_palette = NULL,
  title = NULL
)
```

Arguments

impact_df	Impact data frame from communication_impact().
top_n	Number of top signals to show.
show_sender	Show sender signals.
show_receiver	Show receiver signals.
color_palette	Color palette.
title	Plot title.

Value

A ggplot2 object.

---

plot_spatial_autocorrelation

Plot spatial autocorrelation results

Description

Plot spatial autocorrelation results

Usage

```
plot_spatial_autocorrelation(moran_df, p_threshold = 0.05, title = NULL)
```

Arguments

moran_df	Data frame from communication_spatial_autocorrelation().
p_threshold	P-value threshold for significance.
title	Plot title.

Value

A ggplot2 object.

plotting *Visualization Functions*

Description

Plotting functions for COMMOTR results.

print.commotr_results *Print method for COMMOTR results*

Description

Print method for COMMOTR results

Usage

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

Arguments

x	COMMOTR results object
...	Additional arguments (ignored)

Value

Invisible x

spatial_communication *Spatial Communication Inference*

Description

Infer cell-cell communication in spatial transcriptomics data using collective optimal transport.

Infer cell-cell communication from spatial transcriptomics data using collective optimal transport (COT). The method models ligand-receptor interactions as optimal transport problems with spatial distance constraints.

Usage

```
spatial_communication(
  seurat_obj,
  df_ligrec,
  database_name = "commot",
  assay = NULL,
  slot = "data",
  spatial_coords = NULL,
  dis_thr = 500,
  cost_scale = NULL,
  cost_type = c("euc", "euc_square"),
  heteromeric = TRUE,
  heteromeric_delimiter = "_",
  heteromeric_rule = c("min", "ave"),
  cot_eps_p = 0.1,
  cot_rho = 10,
  cot_eps_mu = NULL,
  cot_eps_nu = NULL,
  cot_strategy = c("pairwise", "collective", "sender", "receiver"),
  cot_weights = c(0.25, 0.25, 0.25, 0.25),
  cot_nitermax = 10000L,
  cot_stopthr = 1e-08,
  pathway_sum = TRUE,
  total_sum = TRUE,
  smooth = FALSE,
  smooth_k = 5L,
  smooth_bandwidth = NULL,
  n_workers = 1L,
  verbose = TRUE
)
```

Arguments

seurat_obj	Seurat object containing spatial transcriptomics data.
df_ligrec	Data frame of ligand-receptor pairs with columns: ligand, receptor, pathway, and optionally signaling_type. Obtain using ligand_receptor_database and filter_lr_database.
database_name	Character string for naming stored results.
assay	Assay to use for expression data (default: DefaultAssay).
slot	Expression slot or layer to use (default: "data").
spatial_coords	Spatial coordinates: NULL (auto-detect), character (reduction name), or matrix (n_cells x 2).
dis_thr	Distance threshold for communication. Cell pairs beyond this distance have zero transport (default: 500).
cost_scale	Scaling factor for cost normalization (default: 1/dis_thr).

`cost_type` Type of distance cost: "euc" (Euclidean distance, default) or "euc_square" (squared Euclidean distance). Using squared distance penalizes longer distances more heavily.

`heteromeric` Logical, whether ligand/receptor can be multi-subunit complexes with subunits separated by delimiter (default: TRUE).

`heteromeric_delimiter` Character separating subunits (default: "_").

`heteromeric_rule` Aggregation rule for complex expression: "min" (default) uses minimum subunit expression, "ave" uses average.

`cot_eps_p` Entropy regularization coefficient (default: 0.1).

`cot_rho` Penalty coefficient for unmatched mass (default: 10).

`cot_eps_mu` Entropy regularization for source marginal (default: eps_p).

`cot_eps_nu` Entropy regularization for target marginal (default: eps_p).

`cot_strategy` COT aggregation strategy:

"collective" Joint optimization (most consistent)

"sender" Optimize per sender signal

"receiver" Optimize per receiver signal

"pairwise" Independent pair optimization (fastest)

`cot_weights` Weights for combining strategies (default: equal).

`cot_nitermax` Maximum Sinkhorn iterations (default: 10000).

`cot_stopthr` Convergence threshold (default: 1e-8).

`pathway_sum` Logical, compute pathway-level aggregates (default: TRUE).

`total_sum` Logical, compute total communication (default: TRUE).

`smooth` Logical, use kernel smoothing for expression (default: FALSE).

`smooth_k` Number of neighbors for smoothing (default: 5).

`smooth_bandwidth` Bandwidth for smoothing (default: NULL, auto).

`n_workers` Number of parallel workers (default: 1).

`verbose` Logical, print progress messages (default: TRUE).

Details

The method solves the unbalanced optimal transport problem:

$$\min_P \langle C, P \rangle + \varepsilon \cdot \mathrm{KL}(P | a \otimes b) + \rho \cdot (\mathrm{KL}(P\mathbf{1}|a) + \mathrm{KL}(P^T\mathbf{1}|b))$$

where C is the cost matrix (spatial distance), a and b are the source (ligand) and target (receptor) distributions, ε is the entropy regularization, and ρ is the penalty for unmatched mass.

Value

Seurat object with COMMOTR results stored in `seurat_obj@misc$commotr[[database_name]]`. Access results using `get_communication_results`.

References

Cang, Z., Zhao, Y., et al. Screening cell-cell communication in spatial transcriptomics via collective optimal transport. Nature Methods 20, 218-228 (2023).

See Also

ligand_receptor_database, filter_lr_database, communication_direction, cluster_communication

Examples

```
## Not run:
# Load ligand-receptor database
df_lr <- ligand_receptor_database("CellChat", "mouse")
df_lr <- filter_lr_database(df_lr, seurat_obj)

# Infer spatial communication
seurat_obj <- spatial_communication(
    seurat_obj,
    df_ligrec = df_lr,
    database_name = "CellChat",
    dis_thr = 500,
    n_workers = 4
)

# Access results
results <- get_communication_results(seurat_obj, "CellChat")

## End(Not run)
```

---

summary.commotr_results

Summary method for COMMOTR results

Description

Summary method for COMMOTR results

Usage

```
## S3 method for class 'commotr_results'
summary(object, ...)
```

Arguments

object	COMMOTR results object
...	Additional arguments (ignored)

Value

Summary data frame

`wasserstein_barycenter` *Compute Wasserstein barycenter with L1 penalty*

Description

Computes the barycenter of multiple distributions using unbalanced optimal transport with L1 (KL divergence) penalty.

Usage

```
wasserstein_barycenter(
  distributions,
  C,
  eps = 0.1,
  m = 10,
  weights = NULL,
  nitermax = 5L
)
```

Arguments

`distributions`	List of distribution vectors
`C`	Cost matrix (n x n)
`eps`	Entropy regularization coefficient
`m`	Penalty for unmatched mass
`weights`	Weights for each distribution (must sum to 1)
`nitermax`	Maximum iterations (default: 5)

Details

This function computes the Wasserstein barycenter, which is the distribution that minimizes the weighted sum of optimal transport distances to all input distributions. The unbalanced formulation allows for mass variation.

Value

Barycenter distribution vector

References

Cang, Z., Zhao, Y., et al. Screening cell-cell communication in spatial transcriptomics via collective optimal transport. Nature Methods 20, 218-228 (2023).

`wasserstein_barycenter_balanced` *Compute regular Wasserstein barycenter*

Description

Computes the barycenter of multiple distributions using standard entropy-regularized optimal transport (balanced).

Usage

```
wasserstein_barycenter_balanced(
  distributions,
  C,
  eps = 0.1,
  weights = NULL,
  nitermax = 10000L
)
```

Arguments

`distributions`	List of distribution vectors (each should sum to 1)
`C`	Cost matrix (n x n)
`eps`	Entropy regularization coefficient
`weights`	Weights for each distribution (must sum to 1)
`nitermax`	Maximum iterations (default: 10000)

Value

Barycenter distribution vector

Index