Package: CrossExpression (via r-universe)

May 25, 2026

Title Cross-Expression Analysis of Spatial Transcriptomics Data

Version 1.0.0

Description Analyzes spatial transcriptomic data using cells-by-genes and cell location matrices to find gene pairs that coordinate their expression between spatially adjacent cells. It enables quantitative analysis and graphical assessment of these cross-expression patterns. See Sarwar et al. (2025) <doi:10.1101/2024.09.17.613579> and <https://github.com/gillislab/CrossExpression/> for more details.

License MIT + file LICENSE

Encoding UTF-8

Imports Rfast, RANN, Matrix, ggplot2, dplyr, stats, stringr

RoxygenNote 7.3.2

NeedsCompilation no

Author Ameer Sarwar [aut, cre], Sarah Choi [ctb], Leon French [ctb], Jesse Gillis [aut]

Maintainer Ameer Sarwar <dogar.ameer@gmail.com>

Depends R (>= 3.5.0)

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

Date/Publication 2025-07-28 19:10:13 UTC

RemoteUrl https://github.com/cran/CrossExpression

RemoteRef HEAD

RemoteSha 7450c48c04aa417d33f26e42fcc5aaa851b07b26

Contents

`bullseye_plot` *Outputs a circular bullseye plot for a gene pair. The central circle is gene B in cells expressing gene A. Rings indicate neighbors with gene B, where the first ring is the first neighbor.*

Description

Outputs a circular bullseye plot for a gene pair. The central circle is gene B in cells expressing gene A. Rings indicate neighbors with gene B, where the first ring is the first neighbor.

Usage

```
bullseye_plot(scores)
```

Arguments

`scores`	Bullseye score as a vector.

Value

Returns a circular bullseye plot.

Examples

```
data("locations")
data("expression")
locations  = as.matrix(locations)
expression = as.matrix(expression)
expression = expression[,1:5]
results    = bullseye_scores(data = expression, locations = locations)
results   = as.numeric(results[1, 3:ncol(results)]) # choose gene pair of interest (row index)
bullseye_plot(results)
```

bullseye_scores *Calculates bullseye statistics for ALL gene pairs. Counts the number of cells with gene B in those with gene A And in their neighbors Neighbor scores are cumulative and normalized by window size.*

Description

Calculates bullseye statistics for ALL gene pairs. Counts the number of cells with gene B in those with gene A And in their neighbors Neighbor scores are cumulative and normalized by window size.

Usage

```
bullseye_scores(
  data,
  locations,
  window_sizes = 1:5,
  ratio_to_co = FALSE,
  log_2 = FALSE,
  output_matrix = FALSE
)
```

Arguments

data	A cells by genes expression matrix.
locations	A cells by coordinates (x-y or higher dimensions) matrix.
window_sizes	Vector of window sizes.
ratio_to_co	If TRUE, results are normalized by co-expressing cells.
log_2	If TRUE, results are transformed by log2.
output_matrix	If TRUE, results are provided in matrix form.

Value

Returns a dataframe with each row representing a gene pair and each column representing the bullseye score for that pair in each window size. output_matrix = TRUE presents a matrix for each window size.

Examples

```
data("locations")
data("expression")
locations  = as.matrix(locations)
expression = as.matrix(expression)
expression = expression[,1:5]
results    = bullseye_scores(data = expression, locations = locations)
```

`correlation` *Computes Pearson's correlation between pairs of columns. If one matrix is provided, the output is the pairwise correlations between its columns. If two matrices are provided, the output is the pairwise correlations between their columns.*

Description

Computes Pearson's correlation between pairs of columns. If one matrix is provided, the output is the pairwise correlations between its columns. If two matrices are provided, the output is the pairwise correlations between their columns.

Usage

```
correlation(matrix1, matrix2 = NULL)
```

Arguments

`matrix1`	The first input matrix.
`matrix2`	The second input matrix (optional).

Value

Returns a correlation matrix.

`cross_expression` *Computes cross-expression and co-expression p-values between all gene pairs.*

Description

Computes cross-expression and co-expression p-values between all gene pairs.

Usage

```
cross_expression(
  data,
  locations,
  neighbor = 1,
  alpha_cross = 0.05,
  alpha_co = 0,
  output_matrix = FALSE
)
```

Arguments

data	A cells by genes expression matrix.
locations	A cells by coordinates (x-y or higher dimensions) matrix.
neighbor	The n-th nearest neighbor for calculating cross-expression.
alpha_cross	The significance threshold for cross-expression.
alpha_co	The significance threshold for co-expression.
output_matrix	If TRUE, outputs the cross-expression p-value matrix.

Value

Returns a list containing gene pairs with co-expression and cross-expression p-values before and after false discovery rate (FDR) correction.

Examples

```
data("locations")
data("expression")
locations  = as.matrix(locations)
expression = as.matrix(expression)
expression = expression[,1:5]
results    = cross_expression(data = expression, locations = locations)
```

---

cross_expression_correlation

Computes gene-gene correlations between cross-expressing cell-neighbor pairs. Cell and neighbor masks are used to consider mutually exclusive expression per gene pair.

Description

Computes gene-gene correlations between cross-expressing cell-neighbor pairs. Cell and neighbor masks are used to consider mutually exclusive expression per gene pair.

Usage

```
cross_expression_correlation(
  data,
  locations,
  neighbor = 1,
  output_matrix = FALSE
)
```

Arguments

data	A cells by genes expression matrix.
locations	A cells by coordinates (x-y or higher dimensions) matrix.
neighbor	The n-th nearest neighbor for computing correlations.
output_matrix	If TRUE, outputs the cross-expression correlation matrix.

Value

Returns a gene list with cross-expression correlation for each gene pair.

Examples

```
data("locations")
data("expression")
locations  = as.matrix(locations)
expression = as.matrix(expression)
expression = expression[,1:5]
results    = cross_expression_correlation(data = expression, locations = locations)
```

---

expression *Example gene expression matrix*

Description

Gene expression matrix for one tissue slice.

Usage

```
data(expression)
```

Format

A matrix with cells as rows and genes as columns.

Source

Example slice obtained from https://data.mendeley.com/datasets/6drcm3hy2h/2

`get_cooccurrence_stats` *Calculates the number of elements common between columns of two matrices. This function performs a simple dot product when binarize = FALSE.*

Description

Calculates the number of elements common between columns of two matrices. This function performs a simple dot product when binarize = FALSE.

Usage

```
get_cooccurrence_stats(matrix1, matrix2, sparse = TRUE, binarize = TRUE)
```

Arguments

`matrix1`	The first input matrix.
`matrix2`	The second input matrix.
`sparse`	If TRUE, coerces matrices to sparse format for efficiency.
`binarize`	If TRUE, converts matrices to binary format to count the number of common elements.

Value

Returns a column by column matrix, where entries represent the number of common elements.

`locations` *Example cell location matrix*

Description

Spatial coordinates for each cell.

Usage

```
data(locations)
```

Format

A matrix with cells as rows and xy-coordinates as columns.

Source

Example slice obtained from https://data.mendeley.com/datasets/6drcm3hy2h/2

`rotate_coordinates` *This function takes x and y coordinates and rotates them counterclockwise by the specified number of degrees, and mean centers the points.*

Description

This function takes x and y coordinates and rotates them counterclockwise by the specified number of degrees, and mean centers the points.

Usage

```
rotate_coordinates(
  x,
  y,
  n_degrees = 0,
  center = FALSE,
  scale = FALSE,
  flip_x = FALSE,
  flip_y = FALSE
)
```

Arguments

x	The x coordinates.
y	The y coordinates.
n_degrees	The degree of rotation in counterclockwise direction.
center	If TRUE, mean centers the points.
scale	If TRUE, standardizes the points.
flip_x	Reflects the points across y = 0 line.
flip_y	Reflects the points across x = 0 line.

Value

Returns a data frame containing the new x and y coordinates.

Examples

```
data("locations")
locations = as.data.frame(as.matrix(locations))
ggplot2::ggplot(data = locations) + ggplot2::aes(x = pos_x, y = pos_y) +
  ggplot2::geom_point(size = 0) + ggplot2::theme_classic()

locations = rotate_coordinates(x = locations$pos_x, y = locations$pos_y, n_degrees = 45)
ggplot2::ggplot(data = locations) + ggplot2::aes(x = pos_x, y = pos_y) +
  ggplot2::geom_point(size = 0) + ggplot2::theme_classic()
```

smooth_cells	*Smooths cells' gene expression by averaging its expression by the nearest neighbors. Optionally computes genes by genes Pearson's correlation matrix between cells by genes and neighbors by genes matrices.*

Description

Smooths cells' gene expression by averaging its expression by the nearest neighbors. Optionally computes genes by genes Pearson's correlation matrix between cells by genes and neighbors by genes matrices.

Usage

```
smooth_cells(data, locations, neighbors_smoothed = 1, corr = TRUE)
```

Arguments

data	A cells by genes expression matrix.
locations	A cells by coordinates (x-y or higher dimensions) matrix.
neighbors_smoothed	Numbers of neighbors used for smoothing (0 is the cell itself; 1 is the nearest neighbor).
corr	If TRUE, computes genes by genes correlation matrix between regions.

Value

Returns a smoothed gene expression matrix. If corr = TRUE, additionally returns a gene-gene correlation matrix.

Examples

```
data("locations")
data("expression")
locations  = as.matrix(locations)
expression = as.matrix(expression)
expression = expression[,1:5]
results    = smooth_cells(data = expression, locations = locations)
```

spatial_enrichment *Determines whether the supplied genes show spatial enrichment in cross-expression. Spatial enrichment can be interpreted as delineating anatomical boundaries.*

Description

Determines whether the supplied genes show spatial enrichment in cross-expression. Spatial enrichment can be interpreted as delineating anatomical boundaries.

Usage

```
spatial_enrichment(
  data,
  locations,
  gene1,
  gene2,
  neighbor = 1,
  max_pairs = 20000
)
```

Arguments

data	A cells by genes expression matrix.
locations	A cells by coordinates (x-y or higher dimensions) matrix.
gene1	Name of gene1.
gene2	Name of gene2.
neighbor	The n-th nearest neighbor.
max_pairs	Specify maximum number of cell pairs to consider. Lower number increases computational efficiency.

Value

Returns a p-value and distance distributions between cross-expressing cells and cross-expressing and random cells.

Examples

```
data("locations")
data("expression")
locations  = as.matrix(locations)
expression = as.matrix(expression)
expression = expression[,1:5]
results    = spatial_enrichment(data = expression, locations = locations,
                                gene1 = "Calb1", gene2 = "Rasgrf2", max_pairs = 100)
```

tissue_expression_plot *Plots gene expression and cross-expression on tissue by coloring cells.*

Description

Plots gene expression and cross-expression on tissue by coloring cells.

Usage

```
tissue_expression_plot(
  data,
  locations,
  gene1,
  gene2,
  cross_expression = TRUE,
  neighbor = 1,
  point_size = 0,
  scale_bar = 0
)
```

Arguments

data	A cells by genes expression matrix.
locations	A cells by coordinates (x-y or higher dimensions) matrix.
gene1	Name of gene 1.
gene2	Name of gene 2.
cross_expression	If TRUE, only cross-expressing cell pairs are shown.
neighbor	The nearest neighbor (for cross-expression).
point_size	Point size on the scatter plot.
scale_bar	Length of the scale bad in microns.

Value

Returns a plot with cells shown as points and color indicating the genes it expresses.

Examples

```
data("locations")
data("expression")
locations  = as.matrix(locations)
expression = as.matrix(expression)
expression = expression[,1:5]
tissue_expression_plot(data = expression, locations = locations, gene1 = "Calb1", gene2 = "Rasgrf2")
```

Index