Package ‘imcRtools’

May 4, 2024

Version 1.10.0

Title Methods for imaging mass cytometry data analysis

Description This R package supports the handling and analysis of imaging mass cytometry and other highly multiplexed imaging data. The main functionality includes reading in single-cell data after image segmentation and measurement, data formatting to perform channel spillover correction and a number of spatial analysis approaches. First, cell-cell interactions are detected via spatial graph construction; these graphs can be visualized with cells representing nodes and interactions representing edges. Furthermore, per cell, its direct neighbours are summarized to allow spatial clustering. Per image/grouping level, interactions between types of cells are counted, averaged and compared against random permutations. In that way, types of cells that interact more (attraction) or less (avoidance) frequently than expected by chance are detected.

License GPL-3

Depends R (>= 4.1), SpatialExperiment

Imports S4Vectors, stats, utils, SummarizedExperiment, methods, pheatmap, scuttle, stringr, readr, EBImage, cytomapper, abind, BiocParallel, viridis, dplyr, magrittr, DT, igraph, SingleCellExperiment, vroom, BiocNeighbors, RTriangle, ggraph, tidygraph, ggplot2, data.table, sf, concaveman, tidyselect, distances, MatrixGenerics, rlang, grDevices

Suggests CATALYST, grid, tidy, BiocStyle, knitr, rmarkdown, markdown, testthat

biocViews ImmunoOncology, SingleCell, Spatial, DataImport, Clustering

VignetteBuilder knitr

URL https://github.com/BodenmillerGroup/imcRtools

BugReports https://github.com/BodenmillerGroup/imcRtools/issues

RoxygenNote 7.2.3

Encoding UTF-8

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

1
git_branch  RELEASE_3_19

git_last_commit  39e011c

git_last_commit_date  2024-04-30

Repository  Bioconductor 3.19

Date/Publication  2024-05-03

Author  Nils Eling [aut],
Tobias Hoch [ctb],
Vito Zanotelli [ctb],
Jana Fischer [ctb],
Daniel Schulz [ctb, cre] (<https://orcid.org/0000-0002-0913-1678>),
Lasse Meyer [ctb]

Maintainer  Daniel Schulz <daniel.schulz@uzh.ch>

Contents

aggregateNeighbors ........................................... 3
binAcrossPixels ............................................. 4
buildSpatialGraph .......................................... 6
countInteractions .......................................... 8
detectCommunity ........................................... 10
detectSpatialContext ...................................... 12
filterPixels .................................................. 15
filterSpatialContext ....................................... 16
findBorderCells ............................................. 18
minDistToCells ............................................. 19
patchDetection ............................................. 21
patchSize ..................................................... 23
plotSpatial ................................................... 24
plotSpatialContext ......................................... 28
plotSpotHeatmap ........................................... 30
readImagefromTXT .......................................... 32
readSCEfromTXT ........................................... 34
read_cpout .................................................. 35
read_steinbock ............................................. 38
show_cpout_features ........................................ 40
testInteractions .......................................... 42

Index  46
aggregateNeighbors

Function to aggregate all neighbors of each cell.

Description

Function to summarize categorical or expression values of all neighbors of each cell.

Usage

aggregateNeighbors(
  object,
  colPairName,
  aggregate_by = c("metadata", "expression"),
  count_by = NULL,
  proportions = TRUE,
  assay_type = NULL,
  subset_row = NULL,
  statistic = c("mean", "median", "sd", "var"),
  name = NULL
)

Arguments

object a SingleCellExperiment or SpatialExperiment object

colPairName single character indicating the colPair(object) entry containing the neighbor information.

aggregate_by character specifying whether the neighborhood should be summarized by cellular features stored in colData(object) (aggregate_by = "metadata") or by marker expression of the neighboring cells (aggregate_by = "expression").

count_by for summarize_by = "metadata", a single character specifying the colData(object) entry containing the cellular metadata that should be summarized across each cell’s neighborhood.

proportions single logical indicating whether aggregated metadata should be returned in form of proportions instead of absolute counts.

assay_type for summarize_by = "expression", single character indicating the assay slot to use.

subset_row for summarize_by = "expression", an integer, logical or character vector specifying the features to use. If NULL, defaults to all features.

statistic for summarize_by = "expression", a single character specifying the statistic to be used for summarizing the expression values across all neighboring cells. Supported entries are "mean", "median", "sd", "var". Defaults to "mean" if not specified.

name single character specifying the name of the data frame to be saved in the colData(object). Defaults to "aggregatedNeighbors" when summarize_by = "metadata" or "statistic_aggregatedExpression" when summarize_by = "expression".
Value

returns an object of class(object) containing the aggregated values in form of a DataFrame object in colData(object)[[name]].

Author(s)

Daniel Schulz (<daniel.schulz@uzh.ch>)

Examples

library(cytomapper)
data(pancreasSCE)
sce <- buildSpatialGraph(pancreasSCE, img_id = "ImageNb",
                       type = "knn", k = 3)

# Aggregating neighboring cell-types
sce <- aggregateNeighbors(sce, colPairName = "knn_interaction_graph",
                          aggregate_by = "metadata",
                          count_by = "CellType")
sce$aggregatedNeighbors

# Aggregating neighboring expression values
sce <- aggregateNeighbors(sce, colPairName = "knn_interaction_graph",
                          aggregate_by = "expression",
                          assay_type = "exprs",
                          statistic = "mean")
sce$mean_aggregatedExpression

---

**binAcrossPixels**

*Aggregate consecutive pixels per single-metal spot*

Description

Helper function for estimating the spillover matrix. Per metal spot, consecutive pixels a aggregated (default: summed).

Usage

```r
binAcrossPixels(
  object,
  bin_size,
  spot_id = "sample_id",
  assay_type = "counts",
  statistic = "sum",
  ...
)
```
binAcrossPixels

Arguments

object a SingleCellExperiment object containing pixel intensities for all channels. Individual pixels are stored as columns and channels are stored as rows.

bin_size single numeric indicating how many consecutive pixels per spot should be aggregated.

spot_id character string indicating which colData(object) entry stores the isotope names of the spotted metal.

assay_type character string indicating which assay to use.

statistic character string indicating the statistic to use for aggregating consecutive pixels.

... additional arguments passed to aggregateAcrossCells

Value

returns the binned pixel intensities in form of a SingleCellExperiment object

Author(s)

Nils Eling (<nils.eling@dqb.m.uzh.ch>)

See Also

aggregateAcrossCells for the aggregation function

Examples

path <- system.file("extdata/spillover", package = "imcRtools")
# Read in .txt files
sce <- readSCEfromTXT(path)
dim(sce)

# Visualizes heatmap before aggregation
plotSpotHeatmap(sce)

# Sum consecutive pixels
sce <- binAcrossPixels(sce, bin_size = 10)
dim(sce)

# Visualizes heatmap after aggregation
plotSpotHeatmap(sce)
buildSpatialGraph  
Builds an interaction graph based on the cells’ locations

Description

Function to define cell-cell interactions via distance-based expansion, delaunay triangulation or k nearest neighbor detection.

Usage

buildSpatialGraph(
  object,
  img_id,
  type = c("expansion", "knn", "delaunay"),
  k = NULL,
  directed = TRUE,
  max_dist = NULL,
  threshold = NULL,
  coords = c("Pos_X", "Pos_Y"),
  name = NULL,
  BNPARAM = KmknnParam(),
  BPPARAM = SerialParam(),
  ...
)

Arguments

object a SingleCellExperiment or SpatialExperiment object
img_id single character indicating the colData(object) entry containing the unique image identifiers.
type single character specifying the type of graph to be build. Supported entries are "expansion" (default) to find interacting cells via distance thresholding; "delaunay" to find interactions via delaunay triangulation; "knn" to find the k nearest neighboring cells.
k (when type = "knn") single numeric integer defining the number of nearest neighbors to search for.
directed (when type = "knn") should the returned graph be directed? (see details).
max_dist (when type = "knn" or type = "delaunay") the maximum distance at which to consider neighboring cells. All neighbors within a distance larger than max_dist will be excluded from graph construction.
threshold (when type = "expansion") single numeric specifying the maximum distance for considering neighbors
coords character vector of length 2 specifying the names of the colData (for a SingleCellExperiment object) or the spatialCoords entries of the cells’ x and y locations.
name single character specifying the name of the graph.
**buildSpatialGraph**

BNPARAM  
a BiocNeighborParam object defining the algorithm to use.

BPPARAM  
a BiocParallelParam-class object defining how to parallelize computations.

...  
additional parameters passed to the findNeighbors function (type = "expansion"),
the triangulate function (type = "delaunay") or the findKNN function (type = "knn").

**Value**

returns a SpatialExperiment or SingleCellExperiment containing the graph in form of a SelfHits object in colPair(object, name). The object is grouped by entries in the img_id slot.

**Building an interaction graph**

This function defines interacting cells in different ways. They are based on the cells’ centroids and do not incorporate cell shape or area.

1. When type = "expansion", all cells within the radius threshold are considered interacting cells.
2. When type = "delaunay", interacting cells are found via a delaunay triangulation of the cells’ centroids.
3. When type = "knn", interacting cells are defined as the k nearest neighbors in the 2D spatial plane.

The directed parameter only affects graph construction via k nearest neighbor search. For directed = FALSE, each interaction will be stored as mutual edge (e.g. node 2 is connected to node 10 and vise versa). For type = "expansion" and type = "delaunay", each edge is stored as mutual edge by default.

The graph is stored in form of a SelfHits object in colPair(object, name). This object can be regarded as an edgelist and coerced to an igraph object via graph_from_edgelist(as.matrix(colPair(object, name))).

**Choosing the graph construction method**

When finding interactions via expansion or knn, the findNeighbors or findKNN functions are used, respectively. Both functions accept the BNPARAM parameter via which the graph construction method can be defined (default KmknnParam). For an overview on the different algorithms, see BiocNeighborParam. Within the BiocNeighborParam object, distance can be set to "Euclidean" (default), "Manhattan" or "Cosine".

**Ordering of the output object**

The buildSpatialGraph function operates on individual images. Therefore the returned object is grouped by entries in img_id. This means all cells of a given image are grouped together in the object. The ordering of cells within each individual image is the same as the ordering of these cells in the input object.

**Author(s)**

Nils Eling (<nils.eling@dqbm.uzh.ch>)
See Also

- `findNeighbors` for the function finding interactions via expansion
- `findKNN` for the function finding interactions via nearest neighbor search
- `triangulate` for the function finding interactions via delaunay triangulation
- `plotSpatial` for visualizing spatial graphs

Examples

```r
path <- system.file("extdata/mockData/steinbock", package = "imcRtools")
spe <- read_steinbock(path)

# Constructing a graph via expansion
spe <- buildSpatialGraph(spe, img_id = "sample_id",
                         type = "expansion", threshold = 10)
colPair(spe, "expansion_interaction_graph")

# Constructing a graph via delaunay triangulation
spe <- buildSpatialGraph(spe, img_id = "sample_id",
                         type = "delaunay")
colPair(spe, "delaunay_interaction_graph")

# Constructing a graph via k nearest neighbor search
spe <- buildSpatialGraph(spe, img_id = "sample_id",
                         type = "knn", k = 5)
colPair(spe, "knn_interaction_graph")
```

**countInteractions**

*Summarizes cell-cell interactions within grouping levels (e.g. images)*

Description

Function to calculate the average number of neighbors B that a cell of type A has using different approaches.

Usage

```r
countInteractions(
  object,
  group_by,
  label,
  colPairName,
  method = c("classic", "histocat", "patch"),
  patch_size = NULL
)
```
countInteractions

Arguments

object  
a SingleCellExperiment or SpatialExperiment object.
group_by  
a single character indicating the colData(object) entry by which interactions are grouped. This is usually the image ID or patient ID.
label  
single character specifying the colData(object) entry which stores the cell labels. These can be cell-types labels or other metadata.
colPairName  
single character indicating the colPair(object) entry containing cell-cell interactions in form of an edge list.
method  
which cell-cell interaction counting method to use (see details)
patch_size  
if method = "patch", a single numeric specifying the minimum number of neighbors of the same type to be considered a patch (see details)

Value

a DataFrame containing one row per group_by entry and unique label entry combination (from_label, to_label). The ct entry stores the interaction count as described in the details. NA is returned if a certain label is not present in this grouping level.

Counting and summarizing cell-cell interactions

In principle, the countInteractions function counts the number of edges (interactions) between each set of unique entries in colData(object)[[label]]. Simplified, it counts for each cell of type A the number of neighbors of type B. This count is averaged within each unique entry colData(object)[[group_by]] in three different ways:

1. method = "classic": The count is divided by the total number of cells of type A. The final count can be interpreted as "How many neighbors of type B does a cell of type A have on average?"
2. method = "histocat": The count is divided by the number of cells of type A that have at least one neighbor of type B. The final count can be interpreted as "How many many neighbors of type B has a cell of type A on average, given it has at least one neighbor of type B?"
3. method = "patch": For each cell, the count is binarized to 0 (less than patch_size neighbors of type B) or 1 (more or equal to patch_size neighbors of type B). The binarized counts are averaged across all cells of type A. The final count can be interpreted as "What fraction of cells of type A have at least a given number of neighbors of type B?"

Author(s)

Vito Zanotelli
Jana Fischer
adapted by Nils Eling (<nils.eling@dqbm.uzh.ch>)

References

See Also

testInteractions for testing cell-cell interactions per grouping level.

Examples

library(cytomapper)
data(pancreasSCE)

pancreasSCE <- buildSpatialGraph(pancreasSCE, img_id = "ImageNb",
                        type = "knn", k = 3)

# Classic style calculation
(out <- countInteractions(pancreasSCE,
                        group_by = "ImageNb",
                        label = "CellType",
                        method = "classic",
                        colPairName = "knn_interaction_graph"))

# Histocat style calculation
(out <- countInteractions(pancreasSCE,
                        group_by = "ImageNb",
                        label = "CellType",
                        method = "histocat",
                        colPairName = "knn_interaction_graph"))

# Patch style calculation
(out <- countInteractions(pancreasSCE,
                        group_by = "ImageNb",
                        label = "CellType",
                        method = "patch",
                        patch_size = 3,
                        colPairName = "knn_interaction_graph"))

---

detectCommunity  

Detect the spatial community of each cell

Description

Function to detect the spatial community of each cell as proposed by Jackson et al., The single-cell pathology landscape of breast cancer, Nature, 2020. Each cell is clustered based on its interactions as defined by a spatial object graph.

Usage

detectCommunity(
    object,
    colPairName,
    size_threshold = 0,
)
detectCommunity

```r
group_by = NULL,
name = "spatial_community",
cluster_fun = "louvain",
BPPARAM = SerialParam()
)
```

**Arguments**

- **object**
  a `SingleCellExperiment` or `SpatialExperiment` object

- **colPairName**
  single character indicating the `colPair(object)` entry containing the neighbor information.

- **size_threshold**
  single positive numeric that specifies the minimum number of cells per community. Defaults to 0.

- **group_by**
  single character indicating that spatial community detection will be performed separately for all unique entries to `colData(object)[,group_by]`.

- **name**
  single character specifying the name of the output saved in `colData(object)`. Defaults to "spatial_community".

- **cluster_fun**
  single character specifying the function to use for community detection. Options are all strings that contain the suffix of an igraph community detection algorithm (e.g. "walktrap"). Defaults to "louvain".

- **BPPARAM**
  a `BiocParallelParam-class` object defining how to parallelize computations. Applicable when `group_by` is specified and defaults to `SerialParam()`. For reproducibility between runs, we recommend defining `RNGseed` in the `BiocParallelParam-class` object.

**Value**

returns an object of class(`object`) containing a new column entry to `colData(object)[[name]]`.

**Spatial community detection procedure**

1. Create an igraph object from the edge list stored in `colPair(object, colPairName)`.
2. Perform community detection using the specified `cluster_fun` algorithm.
3. Store the community IDs in a vector and replace all communities with a size smaller than `size_threshold` by NA.

Optional steps: Specify `group_by` to perform spatial community detection separately for all unique entries to `colData(object)[,group_by]` e.g. for tumor and non-tumor cells.

**Author(s)**

Lasse Meyer (<lasse.meyer@uzh.ch>)

**References**

Jackson et al., The single-cell pathology landscape of breast cancer, Nature, 2020
Examples

```r
library(cytomapper)
library(BiocParallel)
data(pancreasSCE)

sce <- buildSpatialGraph(pancreasSCE, img_id = "ImageNb",
                        type = "expansion",
                        name = "neighborhood",
                        threshold = 20)

## Detect spatial community
set.seed(22)
sce <- detectCommunity(sce,
                        colPairName = "neighborhood",
                        size_threshold = 10)

plotSpatial(sce,
            img_id = "ImageNb",
            node_color_by = "spatial_community",
            scales = "free")

## Detect spatial community - specify group_by
sce <- detectCommunity(sce,
                        colPairName = "neighborhood",
                        group_by = "CellType",
                        size_threshold = 10,
                        BPPARAM = SerialParam(RNGseed = 22))

plotSpatial(sce,
            img_id = "ImageNb",
            node_color_by = "spatial_community",
            scales = "free")
```

detectSpatialContext  
*Detect the spatial context of each cell based on its neighborhood*

Description

Function to detect the spatial context (SC) of each cell. Based on its sorted (high-to-low) cellular neighborhood (CN) fractions in a spatial interaction graph, the SC of each cell is assigned as the set of CNs that cumulatively exceed a user-defined fraction threshold.

The term was coined by Bhate S. et al., Tissue schematics map the specialization of immune tissue motifs and their appropriation by tumors, Cell Systems, 2022 and describes tissue regions in which distinct CNs may be interacting.
detectSpatialContext

Usage

detectSpatialContext(
  object,
  entry = "aggregatedNeighbors",
  threshold = 0.9,
  name = "spatial_context"
)

Arguments

object a SingleCellExperiment or SpatialExperiment object
entry single character specifying the colData(object) entry containing the aggregateNeighbors DataFrame output. Defaults to "aggregatedNeighbors".
threshold single numeric between 0 and 1 that specifies the fraction threshold for SC assignment. Defaults to 0.9.
name single character specifying the name of the output saved in colData(object). Defaults to "spatial_context".

Value

returns an object of class(object) containing a new column entry to colData(object)[[name]]

Spatial context background

The function relies on CN fractions for each cell in a spatial interaction graph (originally a k-nearest neighbor (KNN) graph).

We can retrieve the CN fractions using the buildSpatialGraph and aggregateNeighbors functions.

The window size (k for KNN) for buildSpatialGraph should reflect a length scale on which biological signals can be exchanged and depends, among others, on cell density and tissue area. In view of their divergent functionality, we recommend to use a larger window size for SC (interaction between local processes) than for CN (local processes) detection.

Subsequently, the CN fractions are sorted from high-to-low and the SC of each cell is assigned the minimal combination of SCs that additively surpass a user-defined threshold. The default threshold of 0.9 aims to represent the dominant CNs, hence the most prevalent signals, in a given window.

For more details, please refer to: Bhate S. et al., Tissue schematics map the specialization of immune tissue motifs and their appropriation by tumors, Cell Systems, 2022.

Author(s)

Lasse Meyer (<lasse.meyer@uzh.ch>)

References

Bhate S. et al., Tissue schematics map the specialization of immune tissue motifs and their appropriation by tumors, Cell Systems, 2022
See Also

- `filterSpatialContext` for the function to filter spatial contexts
- `plotSpatialContext` for the function to plot spatial context graphs

Examples

```r
set.seed(22)
library(cytomapper)
data(pancreasSCE)

## 1. Cellular neighborhood (CN)
sce <- buildSpatialGraph(pancreasSCE, img_id = "ImageNb",
                        type = "knn",
                        name = "knn_cn_graph",
                        k = 5)
sce <- aggregateNeighbors(sce, colPairName = "knn_cn_graph",
                          aggregate_by = "metadata",
                          count_by = "CellType",
                          name = "aggregatedCellTypes")
cur_cluster <- kmeans(sce$aggregatedCellTypes, centers = 3)
sce$cellular_neighborhood <- factor(cur_cluster$cluster)

plotSpatial(sce, img_id = "ImageNb",
           colPairName = "knn_cn_graph",
           node_color_by = "cellular_neighborhood",
           scales = "free")

## 2. Spatial context (SC)
sce <- buildSpatialGraph(sce, img_id = "ImageNb",
                        type = "knn",
                        name = "knn_sc_graph",
                        k = 15)
sce <- aggregateNeighbors(sce, colPairName = "knn_sc_graph",
                          aggregate_by = "metadata",
                          count_by = "cellular_neighborhood",
                          name = "aggregatedNeighborhood")

# Detect spatial context
sce <- detectSpatialContext(sce, entry = "aggregatedNeighborhood",
                           threshold = 0.9)

plotSpatial(sce, img_id = "ImageNb",
           colPairName = "knn_sc_graph",
           node_color_by = "spatial_context",
           scales = "free")
```
Description

Helper function for estimating the spillover matrix. After assigning each pixel to a spotted mass, this function will filter incorrectly assigned pixels and remove small pixel sets.

Usage

```r
filterPixels(
  object,
  bc_id = "bc_id",
  spot_mass = "sample_mass",
  minevents = 40,
  correct_pixels = TRUE
)
```

Arguments

- **object**: a `SingleCellExperiment` object containing pixel intensities per channel. Individual pixels are stored as columns and channels are stored as rows.
- **bc_id**: character string indicating which `colData(object)` entry stores the estimated mass.
- **spot_mass**: character string indicating which `colData(object)` entry stores the true isotope mass of the spotted metal.
- **minevents**: single numeric indicating the threshold under which pixel sets are excluded from spillover estimation.
- **correct_pixels**: logical indicating if incorrectly assigned pixels should be excluded from spillover estimation.

Value

returns a `SingleCellExperiment` object in which `colData(object)$bc_id` has been adjusted based on the filter criteria.

Author(s)

Vito Zanotelli, adapted by Nils Eling (<nils.eling@dqbm.uzh.ch>)

Examples

```r
path <- system.file("extdata/spillover", package = "imcRtools")
sce <- readSCEfromTXT(path)
assay(sce, "exprs") <- asinh(counts(sce)/5)

# Pre-process via CATALYST
```
filterSpatialContext

Filter spatial contexts

Description

Function to filter detected spatial contexts (SCs) based on a user-defined threshold for number of group entries and/or cells.

Usage

filterSpatialContext(
  object,
  entry = "spatial_context",
  group_by = "sample_id",
  group_threshold = NULL,
  cells_threshold = NULL,
  name = "spatial_context_filtered"
)

Arguments

object a SingleCellExperiment or SpatialExperiment object
entry a single character specifying the colData(object) entry containing the detectSpatialContext output. Defaults to "spatial_context".
group_by a single character indicating the colData(object) entry by which SCs are grouped. This is usually the image or patient ID. Defaults to "sample_id".
group_threshold a single numeric specifying the minimum number of group entries in which a SC is detected.
cells_threshold a single numeric specifying the minimum total number of cells in a SC.
name a single character specifying the name of the output saved in colData(object). Defaults to "spatial_context_filtered".
Value
returns an object of class(object) containing a new column entry to colData(object)[["name"]]
and a new data.frame entry to metadata(object)["filterSpatialContext"] containing the
group and cell counts per SC.

Author(s)
Lasse Meyer (<lasse.meyer@uzh.ch>)

References
Bhate S. et al., Tissue schematics map the specialization of immune tissue motifs and their appro-
priation by tumors, Cell Systems, 2022

See Also
detectSpatialContext for the function to detect spatial contexts
plotSpatialContext for the function to plot spatial context graphs

Examples
set.seed(22)
library(cytomapper)
data(pancreasSCE)

## 1. Cellular neighborhood (CN)
sce <- buildSpatialGraph(pancreasSCE, img_id = "ImageNb",  
  type = "knn",  
  name = "knn_cn_graph",  
  k = 5)
sce <- aggregateNeighbors(sce, colPairName = "knn_cn_graph",  
  aggregate_by = "metadata",  
  count_by = "CellType",  
  name = "aggregatedCellTypes")
cur_cluster <- kmeans(sce$aggregatedCellTypes, centers = 3)
sce$cellular_neighborhood <- factor(cur_cluster$cluster)
plotSpatial(sce, img_id = "ImageNb",  
  colPairName = "knn_cn_graph",  
  node_color_by = "cellular_neighborhood",  
  scales = "free")

## 2. Spatial context (SC)
sce <- buildSpatialGraph(sce, img_id = "ImageNb",  
  type = "knn",  
  name = "knn_sc_graph",  
  k = 15)
sce <- aggregateNeighbors(sce, colPairName = "knn_sc_graph",  
  aggregate_by = "metadata",  
  count_by = "CellType",  
  name = "aggregatedCellTypes")

cur_cluster <- kmeans(sce$aggregatedCellTypes, centers = 3)
sce$cellular_neighborhood <- factor(cur_cluster$cluster)
plotSpatial(sce, img_id = "ImageNb",  
  colPairName = "knn_sc_graph",  
  node_color_by = "cellular_neighborhood",  
  scales = "free")
findBorderCells

Find cells at the image border

Description
Detection of cells close to the image border for subsequent exclusion from downstream analyses.

Usage
findBorderCells(object, img_id, border_dist, coords = c("Pos_X","Pos_Y"))

Arguments
object  a SingleCellExperiment or SpatialExperiment object.
img_id  single character indicating the colData(object) entry containing the unique image identifiers.
minDistToCells

border_dist single numeric defining the distance to the image border. The image border here is defined as the minimum and maximum among the cells’ x and y location.

coops character vector of length 2 specifying the names of the colData (for a SingleCellExperiment object) or the spatialCoords entries indicating the cells’ x and y locations.

Value

an object of class(object) containing the logical border_cells entry in the colData slot.

Examples

library(cytomapper)
data("pancreasSCE")
sce <- findBorderCells(pancreasSCE, img_id = "ImageNb",
border_dist = 10)

plotSpatial(sce,
    img_id = "ImageNb",
    node_color_by = "border_cells",
    scales = "free")

minDistToCells Function to calculate minimal distance to cells of interest

Description

Function to return the distance of the closest cell of interest for each cell in the data. In the case of patched/clustered cells negative distances are returned by default which indicate the distance of the cells of interest to the closest cell that is not of the type of cells of interest.

Usage

minDistToCells(
    object,
    x_cells,
    img_id,
    name = "distToCells",
    coords = c("Pos_X", "Pos_Y"),
    return_neg = TRUE,
    BPPARAM = SerialParam()
)
Arguments

- **object**
  a SingleCellExperiment or SpatialExperiment object

- **x_cells**
  logical vector of length equal to the number of cells contained in object. TRUE entries define the cells to which distances will be calculated.

- **img_id**
  single character indicating the colData(object) entry containing the unique image identifiers.

- **name**
  character specifying the name of the colData entry to save the distances in.

- **coords**
  character vector of length 2 specifying the names of the colData (for a SingleCellExperiment object) or the spatialCoords entries of the cells’ x and y locations.

- **return_neg**
  logical indicating whether negative distances are to be returned for the distances of patched/spatially clustered cells.

- **BPPARAM**
  a BiocParallelParam-class object defining how to parallelize computations.

Value

returns an object of class(object) containing a new column entry to colData(object)[[name]]. Cells in the object are grouped by entries in img_id.

Ordering of the output object

The minDistToCells function operates on individual images. Therefore the returned object is grouped by entries in img_id. This means all cells of a given image are grouped together in the object. The ordering of cells within each individual image is the same as the ordering of these cells in the input object.

Author(s)

Daniel Schulz (<daniel.schulz@uzh.ch>)

Examples

```r
library(cytomapper)
data(pancreasSCE)

# Build interaction graph
pancreasSCE <- buildSpatialGraph(pancreasSCE, img_id = "ImageNb", type = "expansion", threshold = 20)

# Detect patches of "celltype_B" cells
pancreasSCE <- patchDetection(pancreasSCE, 
    img_id = "ImageNb", 
    patch_cells = pancreasSCE$CellType == "celltype_B", 
    colPairName = "expansion_interaction_graph", 
    min_patch_size = 20, 
    expand_by = 1)

plotSpatial(pancreasSCE, 
    img_id = "ImageNb", 
    ...)
```
patchDetection

Function to detect patches containing defined cell types

Description

Function to detect spatial clusters of defined types of cells. By defining a certain distance threshold, all cells within the vicinity of these clusters are detected as well.

Usage

patchDetection(
  object,
  patch_cells,
  colPairName,
  min_patch_size = 1,
  name = "patch_id",
  expand_by = 0,
  coords = c("Pos_X", "Pos_Y"),
  convex = FALSE,
  img_id = NULL,
  BPPARAM = SerialParam()
)

Arguments

object a SingleCellExperiment or SpatialExperiment object
patch_cells logical vector of length equal to the number of cells contained in object. TRUE entries define the cells to consider for patch detection (see Details).

colPairName single character indicating the colPair(object) entry containing the neighbor information.

min_patch_size single integer indicating the minimum number of connected cells that make up a patch before expansion.
### patchDetection

**name**

Single character specifying the `colData` entry storing the patch IDs in the returned object.

**expand_by**

Single numeric indicating in which vicinity range cells should be considered as belonging to the patch (see Details).

**coords**

Character vector of length 2 specifying the names of the `colData` (for a `SingleCellExperiment` object) or the `spatialCoords` entries of the cells' x and y locations.

**convex**

Should the convex hull be computed before expansion? Default: the concave hull is computed.

**img_id**

Single character indicating the `colData(object)` entry containing the unique image identifiers.

**BPPARAM**

A `BiocParallelParam-class` object defining how to parallelize computations.

#### Value

An object of `class(object)` containing a patch ID for each cell in `colData(object)[[name]]`. If `expand_by > 0`, cells in the output object are grouped by entries in `img_id`.

#### Detecting patches of defined cell types

This function works as follows:

1. Only cells defined by `patch_cells` are considered for patch detection.

2. Patches of connected cells are detected. Here, cell-to-cell connections are defined by the interaction graph stored in `colPair(object, colPairName)`. At this point, patches that contain fewer than `min_patch_size` cells are removed.

3. If `expand_by > 0`, a concave (default) or convex hull is constructed around each patch. This is then expanded by `expand_by` and cells within the expanded hull are detected and assigned to the patch. This expansion only works if a patch contains at least 3 cells.

The returned object contains an additional entry `colData(object)[[name]]`, which stores the patch ID per cell. NA indicate cells that are not part of a patch.

#### Ordering of the output object

If `expand_by > 0`, the `patchDetection` function operates on individual images. Therefore the returned object is grouped by entries in `img_id`. This means all cells of a given image are grouped together in the object. The ordering of cells within each individual image is the same as the ordering of these cells in the input object.

If `expand_by = 0`, the ordering of cells in the output object is the same as in the input object.

#### Author(s)

Tobias Hoch

Adapted by Nils Eling (<nils.eling@dqbm.uzh.ch>)

#### References

patchSize

Examples

```r
library(cytomapper)
data(pancreasSCE)

# Visualize cell types
plotSpatial(pancreasSCE,
            img_id = "ImageNb",
            node_color_by = "CellType",
            scales = "free")

# Build interaction graph
pancreasSCE <- buildSpatialGraph(pancreasSCE,
                                 img_id = "ImageNb",
                                 type = "expansion",
                                 threshold = 20)

# Detect patches of "celltype_B" cells
pancreasSCE <- patchDetection(pancreasSCE,
                              patch_cells = pancreasSCE$CellType == "celltype_B",
                              colPairName = "expansion_interaction_graph")

plotSpatial(pancreasSCE,
            img_id = "ImageNb",
            node_color_by = "patch_id",
            scales = "free")

# Include cells in vicinity
pancreasSCE <- patchDetection(pancreasSCE,
                              patch_cells = pancreasSCE$CellType == "celltype_B",
                              colPairName = "expansion_interaction_graph",
                              expand_by = 20,
                              img_id = "ImageNb")

plotSpatial(pancreasSCE,
            img_id = "ImageNb",
            node_color_by = "patch_id",
            scales = "free")
```

patchSize

Function to compute the area of cell patches

Description

This function constructs polygons around patch cells and computes their area.

Usage

```r
patchSize(
    object,
    patch_name = "patch_id",
    
```
plotSpatial

doctors = c("Pos_X", "Pos_Y"),
convex = FALSE)

Arguments

object a SingleCellExperiment or SpatialExperiment object
patch_name single character indicating the colData(object) entry containing the patch cell identifiers.
coords character vector of length 2 specifying the names of the colData (for a SingleCellExperiment object) or the spatialCoords entries of the cells' x and y locations.
convex should the convex hull be computed to construct the polygon? Default: the concave hull is computed.

Value
A DataFrame object containing the patch identifier, the constructed polygon and the polygon size.

Author(s)
Nils Eling (<nils.eling@dqb.uzh.ch>)

Examples

library(cytomapper)
data(pancreasSCE)

# Build interaction graph
pancreasSCE <- buildSpatialGraph(pancreasSCE, img_id = "ImageNb",
        type = "expansion", threshold = 20)

# Detect patches of "celltype_B" cells
pancreasSCE <- patchDetection(pancreasSCE,
        patch_cells = pancreasSCE$CellType == "celltype_B",
        expand_by = 5, img_id = "ImageNb",
        colPairName = "expansion_interaction_graph")

# Compute the patch area
patchSize(pancreasSCE)

plotSpatial

A general function to plot spatial locations of cells while specifying color, shape, size. Cell-cell interactions can be visualized in form of edges between points.
plotSpatial

Usage

plotSpatial(
  object,
  img_id,
  coords = c("Pos_X", "Pos_Y"),
  node_color_by = NULL,
  node_shape_by = NULL,
  node_size_by = NULL,
  node_color_fix = NULL,
  node_shape_fix = NULL,
  node_size_fix = NULL,
  assay_type = NULL,
  draw_edges = FALSE,
  directed = TRUE,
  edge_color_by = NULL,
  edge_width_by = NULL,
  edge_color_fix = NULL,
  edge_width_fix = NULL,
  arrow = NULL,
  end_cap = NULL,
  colPairName = NULL,
  nodes_first = TRUE,
  ncols = NULL,
  nrows = NULL,
  scales = "fixed",
  flip_x = FALSE,
  flip_y = TRUE,
  aspect_ratio = "auto"
)

Arguments

object a SingleCellExperiment or SpatialExperiment object.

img_id single character indicating the colData(object) entry containing the unique image identifiers.

coords character vector of length 2 specifying the names of the colData (for a SingleCellExperiment object) or the spatialCoords entries indicating the cells' x and y locations.

node_color_by single character indicating the colData(object) entry or marker name by which the nodes (cell locations) should be colored.

node_shape_by single character indicating the colData(object) entry by which the shape of the nodes are defined.

node_size_by single character indicating the colData(object) entry by which the size of the nodes are defined.

node_color_fix single character or numeric specifying the color of all nodes.

node_shape_fix single numeric or character specifying the shape of all nodes.

node_size_fix single numeric specifying the size of all nodes.
plotSpatial

- **assay_type**: single character indicating the assay slot from which to extract the expression data when `node_color_by` is set to one of `rownames(object)`.
- **draw_edges**: should cell-cell interactions be drawn as edges between nodes?
- **directed**: should cell-cell interactions be handled as a directed graph?
- **edge_color_by**: single character indicating by which to color the edges. See details for more information.
- **edge_width_by**: single character determining the size of the edges. See details for more information.
- **edge_color_fix**: single character or numeric specifying the color of all edges.
- **edge_width_fix**: single numeric specifying the size of all edges.
- **arrow**: an `arrow` object specifying how to draw arrows between cells.
- **end_cap**: a `geometry` object specifying how long the edges are. This only takes effect when drawing arrows. Default: `end_cap = circle(0.1, 'cm')`
- **colPairName**: single character specifying the `colPair(object)` slot to retrieve the cell-cell pairings.
- **nodes_first**: should the nodes be plotted first and then the edges?
- **ncols**: number of columns of the grid to arrange individual images.
- **nrows**: number of rows of the grid to arrange individual images.
- **scales**: one of "free", "fixed", "free_x" or "free_y" indicating if x- and y-axis ranges should be fixed across all images. Defaults to "fixed" to match physical units on the x- and y-axis.
- **flip_x**: flip the x-axis?
- **flip_y**: flip the y-axis?
- **aspect_ratio**: single numeric, "auto" or NULL to define the relative ratio between the physical units of the x and y axis. If "auto" (default), the physical units match between the x and y axis if `scales = "fixed"`. If `scales = "free"`, the default aspect ratio is set to 1. Ignore setting the aspect ratio with `aspect_ratio = NULL`.

**Value**

returns a `ggplot` object.

**Visualizing cell locations and cell-cell interactions**

By default, the cells’ locations are visualized in form of points (here also referred to as "nodes") on a 2-dimensional plane. The cells’ coordinates are extracted either from `colData(object)` slot (for a `SingleCellExperiment` input object) or from the `spatialCoords(object)` slot (for a `SpatialExperiment` input object). Node aesthetics are controlled by setting `node_color_by`, `node_shape_by` and `node_size_by` for associating the aesthetics with variables. If node aesthetics should be the same for all nodes, `node_color_fix`, `node_shape_fix` and `node_size_fix` can be set.

When `draw_edges = TRUE`, cell-cell interactions are visualized in form of edges between nodes. For this, `object` needs to contain column pairings in `colPair(object, colPairName)`. Edge color and size can be set by specifying either an entry in `mcols(colPair(object, colPairName))` (edge
attributes) or in colData(object). In the latter case, edges are colored by attributes associated to the "from" node. Variable aesthetics can be set using edge_color_by and edge_width_by. If all edges should have the same width or color, edge_color_fix and edge_width_fix can be set.

Arrows for displaying directed graphs can be drawn by supplying a arrow object. Arrow attributes can be set within this class. To cap the edge before it reaches the next node, the end_cap parameter can be used.

Author(s)

Nils Eling (<nils.eling@dqbm.uzh.ch>)

See Also

buildSpatialGraph for constructing interaction graphs

ggraph for handling graph aesthetics

Examples

library(cytomapper)
data(pancreasSCE)
sce <- buildSpatialGraph(pancreasSCE, img_id = "ImageNb",
 type = "knn", k = 3, directed = FALSE)

# Only nodes
plotSpatial(sce, img_id = "ImageNb",
 node_color_by = "CellType",
 node_shape_by = "ImageNb",
 node_size_by = "Area",
 scales = "free")

# With edges and nodes colored by expression
plotSpatial(sce, img_id = "ImageNb",
 node_color_by = "PIN",
 assay_type = "exprs",
 node_shape_by = "ImageNb",
 node_size_by = "Area",
 draw_edges = TRUE,
 colPairName = "knn_interaction_graph",
 edge_color_by = "Pattern",
 scales = "free")

# With arrows
plotSpatial(sce, img_id = "ImageNb",
 node_color_by = "CellType",
 node_shape_by = "ImageNb",
 node_size_by = "Area",
 draw_edges = TRUE,
 colPairName = "knn_interaction_graph",
 edge_color_fix = "green",
 arrow = grid::arrow(length = grid::unit(0.1, "inch")),

draw_edges = TRUE)
plotSpatialContext

Plot spatial context graph

Description

Function to plot directed spatial context graphs based on symbolic edge-lists and vertex metadata, which operates on the cohort-level. The user can specify node, node_label and edge aesthetics.

Usage

plotSpatialContext(
  object,
  entry = "spatial_context",
  group_by = "sample_id",
  node_color_by = NULL,
  node_size_by = NULL,
  node_color_fix = NULL,
  node_size_fix = NULL,
  node_label_repel = TRUE,
  node_label_color_by = NULL,
  node_label_color_fix = NULL,
  draw_edges = TRUE,
  edge_color_fix = NULL,
  return_data = FALSE
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>a SingleCellExperiment or SpatialExperiment object.</td>
</tr>
<tr>
<td>entry</td>
<td>single character specifying the colData(object) entry containing the detectSpatialContext output. Defaults to &quot;spatial_context&quot;.</td>
</tr>
<tr>
<td>group_by</td>
<td>a single character indicating the colData(object) entry by which SCs are grouped. This is usually the image or patient ID. Defaults to &quot;sample_id&quot;.</td>
</tr>
<tr>
<td>node_color_by</td>
<td>single character either NULL, &quot;name&quot;, &quot;n_cells&quot;, &quot;n_group&quot; by which the nodes should be colored.</td>
</tr>
<tr>
<td>node_size_by</td>
<td>single character either NULL, &quot;n_cells&quot;, &quot;n_group&quot; by which the size of the nodes are defined.</td>
</tr>
<tr>
<td>node_color_fix</td>
<td>single character specifying the color of all nodes.</td>
</tr>
<tr>
<td>node_size_fix</td>
<td>single numeric specifying the size of all nodes.</td>
</tr>
<tr>
<td>node_label_repel</td>
<td>should nodes be labelled? Defaults to TRUE.</td>
</tr>
<tr>
<td>return_data</td>
<td>return a matrix containing the node and edge coordinates if return_data is TRUE.</td>
</tr>
</tbody>
</table>
node_label_color_by
    single character either NULL, "name", "n_cells", "n_group" by which the node labels should be colored.

node_label_color_fix
    single character specifying the color of all node labels.

draw_edges
    should edges be drawn between nodes? Defaults to TRUE.

edge_color_fix
    single character specifying the color of all edges.

return_data
    should the edge list and vertex metadata for graph construction be returned as a list of two data.frames?

Value
    returns a ggplot object or a list of two data.frames.

Author(s)
    Lasse Meyer (<lasse.meyer@uzh.ch>)

References
    Bhate S. et al., Tissue schematics map the specialization of immune tissue motifs and their appropriation by tumors, Cell Systems, 2022

See Also
    detectSpatialContext for the function to detect spatial contexts
    filterSpatialContext for the function to filter spatial contexts

Examples
    set.seed(22)
    library(cytomapper)
    data(pancreasSCE)

    ## 1. Cellular neighborhood (CN)
    sce <- buildSpatialGraph(pancreasSCE, img_id = "ImageNb",
                            type = "knn",
                            name = "knn_cn_graph",
                            k = 5)

    sce <- aggregateNeighbors(sce, colPairName = "knn_cn_graph",
                                aggregate_by = "metadata",
                                count_by = "CellType",
                                name = "aggregatedCellTypes")

    cur_cluster <- kmeans(sce$aggregatedCellTypes, centers = 3)
    sce$cellular_neighborhood <- factor(cur_cluster$cluster)

    plotSpatial(sce, img_id = "ImageNb",
                colPairName = "knn_cn_graph",
## 2. Spatial context (SC)

```r
sce <- buildSpatialGraph(sce, img_id = "ImageNb",
type = "knn",
name = "knn_sc_graph",
k = 15)
```

```r
sce <- aggregateNeighbors(sce, colPairName = "knn_sc_graph",
aggregate_by = "metadata",
count_by = "cellular_neighborhood",
name = "aggregatedNeighborhood")
```

```r
# Detect spatial context
sce <- detectSpatialContext(sce, entry = "aggregatedNeighborhood",
threshold = 0.9)
```

```r
plotSpatial(sce, img_id = "ImageNb",
colPairName = "knn_sc_graph",
node_color_by = "spatial_context",
scales = "free")
```

```r
# Plot spatial context - default
plotSpatialContext(sce, group_by = "ImageNb")
```

```r
# Plot spatial context - adjust aesthetics
plotSpatialContext(sce, group_by = "ImageNb",
node_color_by = "name",
node_size_by = "n_cells",
node_label_color_by = "name")
```

```r
plotSpatialContext(sce, group_by = "ImageNb",
node_color_by = "n_cells",
node_size_by = "n_group")
```

```r
# Plot spatial context - return data
plotSpatialContext(sce, group_by = "ImageNb",
return_data = TRUE)
```

---

**plotSpotHeatmap**

*Summarizes and visualizes the pixel intensities per spot and channel*

### Description

Helper function for estimating the spillover matrix. This function visualizes the median pixel intensities per spot (rows) and per channel (columns) in form of a heatmap.
plotSpotHeatmap

Usage

plotSpotHeatmap(
  object,
  spot_id = "sample_id",
  channel_id = "channel_name",
  assay_type = "counts",
  statistic = "median",
  log = TRUE,
  threshold = NULL,
  order_metals = TRUE,
  color = viridis(100),
  breaks = NA,
  legend_breaks = NA,
  cluster_cols = FALSE,
  cluster_rows = FALSE,
  ...
)

Arguments

object a SingleCellExperiment object containing pixel intensities per channel. Individual pixels are stored as columns and channels are stored as rows.

spot_id character string indicating which colData(object) entry stores the isotope names of the spotted metal. Entries should be of the form (mt)(mass) (e.g. Sm152 for Samarium isotope with the atomic mass 152).

channel_id character string indicating which rowData(object) entry contains the isotope names of the acquired channels.

assay_type character string indicating which assay to use (default counts).

statistic the statistic to use when aggregating channels per spot (default median).

log should the aggregated pixel intensities be log10(x + 1) transformed?

threshold single numeric indicating a threshold after pixel aggregation. All aggregated values larger than threshold will be labeled as 1.

order_metals should the metals be ordered based on spotted mass?

color see parameter in pheatmap

breaks see parameter in pheatmap

legend_breaks see parameter in pheatmap

cluster_cols see parameter in pheatmap

cluster_rows see parameter in pheatmap

... other arguments passed to pheatmap.

Value

a pheatmap object
Quality control for spillover estimation

Visualizing the aggregated pixel intensities serves two purposes:

1. Small median pixel intensities (< 200 counts) might hinder the robust estimation of the channel spillover. In that case, consecutive pixels can be summed (see `binAcrossPixels`).
2. Each spotted metal (row) should show the highest median pixel intensity in its corresponding channel (column). If this is not the case, either the naming of the .txt files was incorrect or the incorrect metal was spotted.

By setting the threshold parameter, the user can easily identify spots where pixel intensities are too low for robust spillover estimation.

Author(s)

Nils Eling (<nils.eling@dqbm.uzh.ch>)

See Also

`pheatmap` for visual modifications

`aggregateAcrossCells` for the aggregation function

Examples

```r
path <- system.file("extdata/spillover", package = "imcRtools")
# Read in .txt files
sce <- readSCEfromTXT(path)

# Visualizes heatmap
plotSpotHeatmap(sce)

# Visualizes thresholding results
plotSpotHeatmap(sce, log = FALSE, threshold = 200)
```

readImagefromTXT

Reads one or multiple .txt files into a CytoImageList object

Description

Reader function to generate `Image` objects in form of a `CytoImageList` container from .txt files.

Usage

```r
readImagefromTXT(
  path,
  pattern = ".txt$",
  channel_pattern = "[A-Za-z]{1,2}[0-9]{2,3}Di",
  index_names = c("X", "Y"),
  BPPARAM = SerialParam()
)
```
**Arguments**

- **path**: Full path to where the individual .txt files are located. This is usually the path where the .mcd file is located.
- **pattern**: pattern to select which files should be read in (default ".txt$").
- **channel_pattern**: regular expression to select the channel names from the files.
- **index_names**: exact names of the columns storing the x and y coordinates of the image
- **BPPARAM**: parameters for parallelized reading in of images. This is only recommended for very large images.

**Value**

returns a **CytoImageList** object containing one **Image** object per .txt file.

**Imaging mass cytometry .txt files**

As part of the raw data folder, the Hyperion imaging system writes out one .txt file per acquisition. These files store the ion counts per pixel and channel.

This function reads these .txt files into a single **CytoImageList** object for downstream analysis. The **pattern** argument allows selection of all .txt files or a specific subset of files. The **channelNames** of the CytoImageList object are determined by the **channel_pattern** argument.

**Author(s)**

Nils Eling (<nils.eling@dqbm.uzh.ch>)

**See Also**

- **CytoImageList** for the container
- **MulticoreParam** for parallelized processing
- **Image** for the multi-channel image object
- vignette("cytomapper") for visualization of multi-channel images

**Examples**

```r
path <- system.file("extdata/mockData/raw", package = "imcRtools")

# Read in all images
x <- readImagefromTXT(path)
x

# Read in specific files
y <- readImagefromTXT(path, pattern = "ROI_002")
y

# Read in other channelNames
z <- readImagefromTXT(path, channel_pattern = "[A-Za-z](\[0-9\](3))")
z
```
readSCEfromTXT  
Generates a SingleCellExperiment from .txt files

Description

Helper function to process raw .txt files acquired by the Hyperion imaging system into a SingleCellExperiment object. This function is mainly used to read-in data generated from a "spillover slide". Here, each .txt file contains the measurements of multiple pixels for a single stain across all open channels.

Usage

readSCEfromTXT(
  x, 
  pattern = ".*\$", 
  metadata_cols = c("Start_push", "End_push", "Pushes_duration", "X", "Y", "Z"), 
  verbose = TRUE, 
  read_metal_from_filename = TRUE
)

Arguments

x  
input can be of different types:

A path  
Full path to where the single stain .txt files are located.

A list object  
A named list object where each entry is a data.frame or coercible to one. The names of each entry indicate the spotted metals (see details).

pattern  
pattern to select which files should be read in (default ".*\$`). Only used when x is a path.

metadata_cols  
character vector indicating which column entries of the .txt files should be saved in the colData(sce) slot.

verbose  
logical indicating if additional information regarding the spotted and acquired masses should be shown.

read_metal_from_filename  
should the sample metal and mass be extracted from the file/object names?

Value

returns a SCE object where pixels are stored as columns and acquired channels are stored as rows.

Reading in .txt files for spillover correction

As described in the original publication, single metal spots are acquired using the Hyperion imaging system. Each acquisition corresponds to one spot. All acquisitions are stored in a single .mcd file and individual acquisitions are stored in single .txt files.

This function aggregates these measurements into a single SingleCellExperiment object. For this, two inputs are possible:
1. x is a path: By default all .txt files are read in from the specified path. Here, the path should indicate the location of the spillover slide measurement. The file names of the .txt file must contain the spotted metal isotope name in the format (mt)(mass) (e.g. Sm152 for Samarium isotope with the atomic mass 152). Internally, the last occurrence of such a pattern is read in as the metal isotope name and stored in the colData(sce)$sample_id slot.

2. x is a named list: If there are issues with reading in the metal isotope names from the .txt file names, the user can provide a list for which each entry contains the contents of a single .txt file. The names of the list must indicate the spotted metal in the format (mt)(mass). These names will be stored in the colData(sce)$sample_id slot.

When read_metal_from_filename = FALSE, the function will not attempt to read in the spotted metal isotopes from the file or list names. Therefore, only the sample_id will be set based on the file/list names.

Author(s)

Nils Eling (<nils.eling@dqbm.uzh.ch>)

References


Examples

```r
# Read files from path
path <- system.file("extdata/spillover", package = "imcRtools")

sce <- readSCEfromTXT(path)
sce

# Read files as list
cur_file_names <- list.files(path, pattern = ".txt", full.names = TRUE)
cur_files <- lapply(cur_file_names, read.delim)
names(cur_files) <- sub(".txt", "", basename(cur_file_names))

sce <- readSCEfromTXT(cur_files)
sce
```

---

**read_cpout**

**Reads in single-cell data generated by the ImcSegmentationPipeline**

**Description**

Reader function to generate a SpatialExperiment or SingleCellExperiment object from single-cell data obtained by the ImcSegmentationPipeline pipeline.
Usage

```r
read_cpout(
  path,
  object_file = "cell.csv",
  image_file = "Image.csv",
  panel_file = "panel.csv",
  graph_file = "Object relationships.csv",
  object_feature_file = "var_cell.csv",
  intensities = "Intensity_MeanIntensity_FullStack",
  extract_imgid_from = "ImageNumber",
  extract_cellid_from = "ObjectNumber",
  extract_coords_from = c("Location_Center_X", "Location_Center_Y"),
  extract_cellmetadata_from = c("AreaShape_Area", "Neighbors_NumberOfNeighbors_8",
    "AreaShape_Eccentricity", "AreaShape_MajorAxisLength", "AreaShape_MinorAxisLength",
    "AreaShape_MeanRadius"),
  extract_imagemetadata_from = c("Metadata_acname", "Metadata_acid",
    "Metadata_description"),
  extract_graphimageid_from = "First Image Number",
  extract_graphcellids_from = c("First Object Number", "Second Object Number"),
  extract_metal_from = "Metal Tag",
  scale_intensities = TRUE,
  extract_scalingfactor_from = "Scaling_FullStack",
  return_as = c("spe", "sce")
)
```

Arguments

- **path**: full path to the CellProfiler output folder.
- **object_file**: single character indicating the file name storing the object/cell-specific intensities and metadata.
- **image_file**: single character indicating the file name storing meta data per image (can be NULL).
- **panel_file**: single character indicating the file name storing the panel information (can be NULL).
- **graph_file**: single character indicating the file name storing the object/cell interaction information (can be NULL).
- **object_feature_file**: single character indicating the file name storing object/cell feature information.
- **intensities**: single character indicating which column entries of the object_file contain the intensity features of interest. See details.
- **extract_imgid_from**: single character indicating which column entries of the object_file and image_file contain the image integer ID.
- **extract_cellid_from**: single character indicating which column entry of the object_file contains the object/cell integer ID.
read_cpout

extract_coords_from
character vector indicating which column entries of the object_file contain the x and y location of the objects/cells.

extract_cellmetadata_from
character vector indicating which additional object/cell specific metadata to extract from the object_file.

extract_imagemetadata_from
character vector indicating which additional image specific metadata to extract from the image_file. These will be stored in the colData(x) slot as object/cell-specific entries.

extract_graphimageid_from
single character indicating which column entries of the graph_file contain the image integer ID.

extract_graphcellids_from
character vector indicating which column entries of the graph_file contain the first and second object/cell integer IDs. These will be stored as the from and to entry of the SelfHits object in colPair(x, "neighborhood").

extract_metal_from
single character indicating which column entry of the panel_file contains the metal isotopes of the used antibodies. This entry is used to match the panel information to the acquired channel information.

scale_intensities
single logical. Should the measured intensity features be scaled by extract_scalingfactor_from.

extract_scalingfactor_from
single character indicating which column entries of the image_file contain the image specific scaling factor.

return_as
should the object be returned as SpatialExperiment (return_as = "spe") or SingleCellExperiment (return_as = "sce").

Value
returns a SpatialExperiment or SingleCellExperiment object with markers in rows and cells in columns.

The returned data container

In the case of both containers x, intensity features (as selected by the intensities parameter) are stored in the counts(x) slot. Cell metadata (e.g. morphological features) are stored in the colData(x) slot. The interaction graphs are stored as SelfHits object in the colPair(x, "neighborhood") slot.

Intensity features are extracted via partial string matching. Internally, the read_cpout function checks if per channel a single intensity feature is read in (by checking the _cXY ending where XY is the channel number).

In the case of a returned SpatialExperiment object, the cell coordinates are stored in the spatialCoords(x) slot.

In the case of a returned SingleCellExperiment object, the cell coordinates are stored in the colData(x) slot named as Pos_X and Pos_Y.
read_steinbock

Author(s)
Tobias Hoch
Nils Eling (<nils.eling@dqbm.uzh.ch>)

See Also

https://github.com/BodenmillerGroup/ImcSegmentationPipeline for the pipeline
read_steinbock for reading in single-cell data as produced by the steinbock pipeline
colPair for information on how to work with the cell-cell interaction graphs

Examples

path <- system.file("extdata/mockData/cpout", package = "imcRtools")

# Read in as SpatialExperiment object
x <- read_cpout(path, graph_file = "Object_relationships.csv")

# Read in as SingleCellExperiment object
x <- read_cpout(path, graph_file = "Object_relationships.csv",
                   return_as = "sce")

Description

Reader function to generate a SpatialExperiment or SingleCellExperiment object from single-cell data obtained by the steinbock pipeline.

Usage

read_steinbock(
  path,
  intensities_folder = "intensities",
  regionprops_folder = "regionprops",
  graphs_folder = "neighbors",
  pattern = NULL,
  extract_cellid_from = "Object",
  extract_coords_from = c("centroid-1", "centroid-0"),
  image_file = "images.csv",
  extract_imagemetadata_from = c("width_px", "height_px"),
  panel_file = "panel.csv",
  extract_names_from = "name",
  return_as = c("spe", "sce"),
)
BPPARAM = SerialParam()
)

Arguments

path                  full path to the steinbock output folder
intensities_folder    name of the folder containing the intensity measurements per image
regionprops_folder    name of the folder containing the cell-specific morphology and spatial measurements per image. Can be set to NULL to exclude reading in morphology measures.
graphs_folder         name of the folder containing the spatial connectivity graphs per image. Can be set to NULL to exclude reading in graphs.
pattern               regular expression specifying a subset of files that should be read in.
extrack_cellid_from   single character indicating which column entry in the intensity files contains the integer cell id.
extrack_coords_from   character vector indicating which column entries in the regionprops files contain the x (first entry) and y (second entry) coordinates.
image_file            single character indicating the file name storing meta data per image (can be NULL).
extrack_imagemetadata_from character vector indicating which additional image specific metadata to extract from the image_file. These will be stored in the colData(x) slot as object/cell-specific entries.
panel_file            single character containing the name of the panel file. This can either be inside the steinbock path (recommended) or located somewhere else.
extrack_names_from    single character indicating the column of the panel file containing the channel names.
return_as             should the object be returned as SpatialExperiment (return_as = "spe") or SingleCellExperiment (return_as = "sce").
BPPARAM               parameters for parallelised processing.

Value

returns a SpatialExperiment or SingleCellExperiment object markers in rows and cells in columns.

The returned data container

In the case of both containers x, intensity features are stored in the counts(x) slot. Morphological features are stored in the colData(x) slot. The graphs are stored as SelfHits object in the colPair(x, "neighborhood") slot.
In the case of a returned SpatialExperiment object, the cell coordinates are stored in the spatialCoords(x) slot.
In the case of a returned SingleCellExperiment object, the cell coordinates are stored in the colData(x) slot named as Pos_X and Pos_Y.

Author(s)

Nils Eling (<nils.eling@dqbm.uzh.ch>)

See Also

https://github.com/BodenmillerGroup/steinbock for the pipeline
read_cpout for reading in single-cell data as produced by the ImcSegmentationPipeline
SingleCellExperiment and SpatialExperiment for the constructor functions.
colPair for information on how to work with the cell-cell interaction graphs
bpparam for the parallelised backend

Examples

path <- system.file("extdata/mockData/steinbock", package = "imcRtools")

# Read in as SpatialExperiment object
x <- read_steinbock(path)
x

# Read in as SingleCellExperiment object
x <- read_steinbock(path, return_as = "sce")
x

# Read in a subset of files
x <- read_steinbock(path, pattern = "mockData1")
x

# Only read in intensities
x <- read_steinbock(path, graphs_folder = NULL, regionprops_folder = NULL)
x

# Parallelisation
x <- read_steinbock(path, BPPARAM = BiocParallel::bpparam())
show_cpout_features

Usage

show_cpout_features(
  path,
  display = c("cell_features", "image_features"),
  cell_features = "var_cell.csv",
  image_features = "var_Image.csv"
)

Arguments

path full path to the CellProfiler output folder

display single character indicating which features to display. Accepted entries are cell_features to display extracted single-cell features or image_features to display extracted image-level features.

cell_features single character indicating the name of the file storing the extracted cell features.

image_features single character indicating the name of the file storing the extracted image features.

Value

da datatable object

Author(s)

Nils Eling (<nils.eling@dqbm.uzh.ch>)

See Also

read_cpout for the CellProfiler reader function

Examples

path <- system.file("extdata/mockData/cpout", package = "imcRtools")

# Display cell features
show_cpout_features(path)

# Display image features
show_cpout_features(path, display = "image_features")
testInteractions  

Tests if cell types interact more or less frequently than random

Description

Cell-cell interactions are summarized in different ways and the resulting count is compared to a distribution of counts arising from random permutations.

Usage

testInteractions(
  object,
  group_by,
  label,
  colPairName,
  method = c("classic", "histocat", "patch"),
  patch_size = NULL,
  iter = 1000,
  p_threshold = 0.01,
  return_samples = FALSE,
  tolerance = sqrt(.Machine$double.eps),
  BPPARAM = SerialParam()
)

Arguments

object a SingleCellExperiment or SpatialExperiment object.
group_by a single character indicating the colData(object) entry by which interactions are grouped. This is usually the image ID or patient ID.
label single character specifying the colData(object) entry which stores the cell labels. These can be cell-types labels or other metadata entries.
colPairName single character indicating the colPair(object) entry containing cell-cell interactions in form of an edge list.
method which cell-cell interaction counting method to use (see details)
patch_size if method = "patch", a single numeric specifying the minimum number of neighbors of the same type to be considered a patch (see details)
iter single numeric specifying the number of permutations to perform
p_threshold single numeric indicating the empirical p-value threshold at which interactions are considered to be significantly enriched or depleted per group.
return_samples single logical indicating if the permuted interaction counts of all iterations should be returned.
tolerance single numeric larger than 0. This parameter defines the difference between the permuted count and the actual counts at which both are regarded as equal. Default taken from all.equal.
BPPARAM parameters for parallelized processing.
**Value**

A DataFrame containing one row per `group_by` entry and unique `label` entry combination (`from_label`, `to_label`). The object contains the following entries:

- `ct`: stores the interaction count as described in the details.
- `p_gt`: stores the fraction of perturbations equal or greater than `ct`.
- `p_lt`: stores the fraction of perturbations equal or less than `ct`.
- `interaction`: is there the tendency for a positive interaction (attraction) between `from_label` and `to_label`? Is `p_lt` greater than `p_gt`?
- `p`: the smaller value of `p_gt` and `p_lt`.
- `sig`: is `p` smaller than `p_threshold`?
- `sigval`: Combination of interaction and `sig`.
  - `-1`: interaction == FALSE and `sig` == TRUE
  - `0`: `sig` == FALSE
  - `1`: interaction == TRUE and `sig` == TRUE

NA is returned if a certain label is not present in this grouping level.

**Counting and summarizing cell-cell interactions**

In principle, the `countInteractions` function counts the number of edges (interactions) between each set of unique entries in `colData(object)[[label]]`. Simplified, it counts for each cell of type A the number of neighbors of type B. This count is averaged within each unique entry `colData(object)[[group_by]]` in three different ways:

1. `method = "classic"`: The count is divided by the total number of cells of type A. The final count can be interpreted as "How many neighbors of type B does a cell of type A have on average?"
2. `method = "histocat"`: The count is divided by the number of cells of type A that have at least one neighbor of type B. The final count can be interpreted as "How many many neighbors of type B has a cell of type A on average, given it has at least one neighbor of type B?"
3. `method = "patch"`: For each cell, the count is binarized to 0 (less than `patch_size` neighbors of type B) or 1 (more or equal to `patch_size` neighbors of type B). The binarized counts are averaged across all cells of type A. The final count can be interpreted as "What fraction of cells of type A have at least a given number of neighbors of type B?"

**Testing for significance**

Within each unique entry to `colData(object)[[group_by]]`, the entries of `colData(object)[[label]]` are randomized `iter` times. For each iteration, the interactions are counted as described above. The result is a distribution of the interaction count under spatial randomness. The observed interaction count is compared against this Null distribution to derive empirical p-values:

- `p_gt`: fraction of perturbations equal or greater than the observed count
- `p_lt`: fraction of perturbations equal or less than the observed count

Based on these empirical p-values, the interaction score (attraction or avoidance), overall p value and significance by comparison to `p_threshold` (`sig` and `sigval`) are derived.
Author(s)

Vito Zanotelli
Jana Fischer
adapted by Nils Eling (<nils.eling@dqbm.uzh.ch>)

References


See Also

countInteractions for counting (but not testing) cell-cell interactions per grouping level.
bpparam for the parallelised backend

Examples

library(cytomapper)
library(BiocParallel)
data(pancreasSCE)

pancreasSCE <- buildSpatialGraph(pancreasSCE, img_id = "ImageNb",
                                  type = "knn", k = 3)

# Classic style calculation - setting the seed inside SerialParam for reproducibility
(out <- testInteractions(pancreasSCE,
                     group_by = "ImageNb",
                     label = "CellType",
                     method = "classic",
                     colPairName = "knn_interaction_graph",
                     iter = 1000,
                     BPPARAM = SerialParam(RNGseed = 123)))

# Histocat style calculation
(out <- testInteractions(pancreasSCE,
                     group_by = "ImageNb",
                     label = "CellType",
                     method = "histocat",
                     colPairName = "knn_interaction_graph",
                     iter = 1000,
                     BPPARAM = SerialParam(RNGseed = 123)))

# Patch style calculation
(out <- testInteractions(pancreasSCE,
                     group_by = "ImageNb",
                     label = "CellType",
                     method = "patch",
                     patch_size = 3,
colPairName = "knn_interaction_graph",
iter = 1000,
BPPARAM = SerialParam(RNGseed = 123))
Index

aggregateAcrossCells, 5, 32
aggregateNeighbors, 3, 13
arrow, 26, 27
binAcrossPixels, 4, 32
BiocNeighborParam, 7
bpparam, 40, 44
buildSpatialGraph, 6, 13, 27
channelNames, 33
colPair, 38, 40
countInteractions, 8, 43, 44
CytoImageList, 32, 33
datatable, 41
detectCommunity, 10
detectSpatialContext, 12, 17, 28, 29
filterPixels, 15
filterSpatialContext, 14, 16, 29
findBorderCells, 18
findKNN, 7, 8
findNeighbors, 7, 8
geometry, 26
ggraph, 27
Image, 32, 33
KmknnParam, 7
minDistToCells, 19
MulticoreParam, 33
patchDetection, 21
patchSize, 23
pheatmap, 31, 32
plotSpatial, 8, 24
plotSpatialContext, 14, 17, 28
plotSpotHeatmap, 30
read_cpout, 35, 40, 41
read_steinbock, 38, 38
readImagefromTXT, 32
readSCEfromTXT, 34
SelfHits, 37, 39
show_cpout_features, 40
SingleCellExperiment, 34, 35, 37–40
SpatialExperiment, 35, 37–40
testInteractions, 10, 42
triangulate, 7, 8

46