

# Package ‘SPIAT’

December 8, 2023

**Type** Package

**Title** Spatial Image Analysis of Tissues

**Version** 1.4.1

**Description** SPIAT (**S**p**a**tial **I**mage **A**nalysis of **T**issues) is an R package with a suite of data processing, quality control, visualization and data analysis tools. SPIAT is compatible with data generated from single-cell spatial proteomics platforms (e.g. OPAL, CODEX, MIBI, cellprofiler). SPIAT reads spatial data in the form of X and Y coordinates of cells, marker intensities and cell phenotypes. SPIAT includes six analysis modules that allow visualization, calculation of cell colocalization, categorization of the immune microenvironment relative to tumor areas, analysis of cellular neighborhoods, and the quantification of spatial heterogeneity, providing a comprehensive toolkit for spatial data analysis.

**License** Artistic-2.0 + file LICENSE

**Encoding** UTF-8

**Depends** R (>= 4.2.0), SpatialExperiment (>= 1.8.0)

**Imports** apcluster (>= 1.4.7), ggplot2 (>= 3.2.1), gridExtra (>= 2.3), gtools (>= 3.8.1), reshape2 (>= 1.4.3), dplyr (>= 0.8.3), RANN (>= 2.6.1), pracma (>= 2.2.5), dbscan (>= 1.1-5), mmand (>= 1.5.4), tibble (>= 2.1.3), grDevices, stats, utils, vroom, dittoSeq, spatstat.geom, methods, spatstat.explore, raster, sp, SummarizedExperiment, rlang

**Suggests** BiocStyle, plotly (>= 4.9.0), knitr, rmarkdown, pkgdown, testthat, graphics, alphahull, Rtsne, umap, ComplexHeatmap, elsa

**biocViews** BiomedicalInformatics, CellBiology, Spatial, Clustering, DataImport, ImmunoOncology, QualityControl, SingleCell, Software, Visualization

**BugReports** <https://github.com/trigosteam/SPIAT/issues>

**RoxygenNote** 7.2.3

**LazyData** true

**VignetteBuilder** knitr

**URL** <https://trigosteam.github.io/SPIAT/>

**git\_url** <https://git.bioconductor.org/packages/SPIAT>

**git\_branch** RELEASE\_3\_18

**git\_last\_commit** 628d15c

**git\_last\_commit\_date** 2023-11-02

**Repository** Bioconductor 3.18

**Date/Publication** 2023-12-07

**Author** Anna Trigos [aut] (<<https://orcid.org/0000-0002-5915-2952>>),  
 Yuzhou Feng [aut, cre] (<<https://orcid.org/0000-0002-2955-4053>>),  
 Tianpei Yang [aut],  
 Mabel Li [aut],  
 John Zhu [aut],  
 Volkan Ozcoban [aut],  
 Maria Doyle [aut]

**Maintainer** Yuzhou Feng <yuzhou.feng@petermac.org>

## R topics documented:

AUC_of_cross_function . . . . .	3
average_marker_intensity_within_radius . . . . .	4
average_minimum_distance . . . . .	5
average_nearest_neighbor_index . . . . .	5
average_percentage_of_cells_within_radius . . . . .	6
calculate_cell_proportions . . . . .	7
calculate_cross_functions . . . . .	8
calculate_distance_to_margin . . . . .	9
calculate_entropy . . . . .	10
calculate_minimum_distances_between_celltypes . . . . .	11
calculate_pairwise_distances_between_celltypes . . . . .	11
calculate_percentage_of_grids . . . . .	12
calculate_proportions_of_cells_in_structure . . . . .	13
calculate_spatial_autocorrelation . . . . .	14
calculate_summary_distances_between_celltypes . . . . .	14
calculate_summary_distances_of_cells_to_borders . . . . .	15
composition_of_neighborhoods . . . . .	16
compute_gradient . . . . .	17
crossing_of_crossK . . . . .	17
defined_image . . . . .	18
define_celltypes . . . . .	19
define_structure . . . . .	20
dimensionality_reduction_plot . . . . .	21
entropy_gradient_aggregated . . . . .	22
format_cellprofiler_to_spe . . . . .	23
format_codex_to_spe . . . . .	24
format_colData_to_spe . . . . .	25

format_halo_to_spe . . . . .	25
format_image_to_spe . . . . .	27
format_inform_to_spe . . . . .	29
format_spe_to_ppp . . . . .	30
grid_metrics . . . . .	31
identify_bordering_cells . . . . .	31
identify_neighborhoods . . . . .	32
image_no_markers . . . . .	33
image_splitter . . . . .	34
marker_intensity_boxplot . . . . .	35
marker_prediction_plot . . . . .	36
marker_surface_plot . . . . .	36
marker_surface_plot_stack . . . . .	37
measure_association_to_cell_properties . . . . .	38
mixing_score_summary . . . . .	39
number_of_cells_within_radius . . . . .	40
plot_average_intensity . . . . .	41
plot_cell_categories . . . . .	42
plot_cell_distances_violin . . . . .	43
plot_cell_marker_levels . . . . .	44
plot_cell_percentages . . . . .	44
plot_composition_heatmap . . . . .	45
plot_distance_heatmap . . . . .	46
plot_marker_level_heatmap . . . . .	47
predict_phenotypes . . . . .	47
R_BC . . . . .	49
select_celltypes . . . . .	50
simulated_image . . . . .	51
SPIAT . . . . .	51

## Index 52

---

AUC\_of\_cross\_function *The difference in AUC of the cross function curves*

---

### Description

Calculate the difference of area under the curve (AUC) between two curves, normalised by the total area of the graph.

### Usage

```
AUC_of_cross_function(df.cross)
```

### Arguments

`df.cross` Data.frame. The output of `calculate_cross_functions`. Containing the positions of the two curves. Columns contain "r", "border" and "theo".

**Value**

A number

**Examples**

```
df_cross <- calculate_cross_functions(SPIAT::defined_image, method = "Kcross",
  cell_types_of_interest = c("Tumour", "Immune3"),
  feature_colname = "Cell.Type", dist = 100)
AUC_of_cross_function(df_cross)
```

---

```
average_marker_intensity_within_radius
  average_marker_intensity_within_radius
```

---

**Description**

Calculates the average intensity of the target\_marker within a radius from the cells positive for the reference marker. Note that it pools all cells with the target marker that are within the specific radius of any reference cell. Results represent the average intensities within a radius, but not a vector of metrics for each cell.

**Usage**

```
average_marker_intensity_within_radius(
  spe_object,
  reference_marker,
  target_marker,
  radius = 20
)
```

**Arguments**

spe_object	SpatialExperiment object in the form of the output of <a href="#">format_image_to_spe</a> .
reference_marker	String specifying the marker that is used for reference cells.
target_marker	String specifying the marker to calculate its average intensity.
radius	Numeric specifying the radius of search for cells around the reference cells.

**Value**

A single number is returned

**Examples**

```
average_marker_intensity_within_radius(SPIAT::simulated_image,
  reference_marker = "Immune_marker3",
  target_marker = "Immune_marker2",
  radius=30)
```

---

```
average_minimum_distance
    average_minimum_distance
```

---

**Description**

Calculates the average minimum distance of all cells to their nearest cells in the input image.

**Usage**

```
average_minimum_distance(spe_object)
```

**Arguments**

spe\_object      SpatialExperiment object in the form of the output of [format\\_image\\_to\\_spe](#).

**Value**

A single number is returned

**Examples**

```
average_minimum_distance(SPIAT::simulated_image)
```

---

```
average_nearest_neighbor_index
    Average nearest neighbor index for point pattern (clustering or dispersion)
```

---

**Description**

Calculate the the average nearest neighbor (ANN) index of a specified type of cells. The index indicates the clustering effect of a point pattern. The pattern can be clustering, random or dispersion.

**Usage**

```
average_nearest_neighbor_index(
  spe_object,
  reference_celltypes,
  feature_colname,
  p_val = 5e-06
)
```

**Arguments**

<code>spe_object</code>	SpatialExperiment object in the form of the output of <code>format_image_to_spe</code> .
<code>reference_celltypes</code>	String Vector. Cells with these cell types will be used for ANNI calculation.
<code>feature_colname</code>	String. Specify the selected column for 'reference_celltypes'.
<code>p_val</code>	Numeric. The p value threshold to determine the significance of a pattern.

**Details**

ANN index is a statistical test to test for the presence of clusters of cells, (Clark and Evans, 1954). The ANN index evaluates the spatial aggregation or dispersion effect of objects based on the average distances between pairs of the nearest objects and can be used to test for the clustering of specific cell types (e.g. immune or tumor cells). Next, the z score and p-value of the ANN index is calculated to validate the significance of the pattern.

**Value**

A list with the ANN index, the pattern type and the corresponding p value

**Examples**

```
average_nearest_neighbor_index(SPIAT::defined_image, reference_celltypes =
  "Tumour", feature_colname = "Cell.Type")
```

---

```
average_percentage_of_cells_within_radius
  average_percentage_of_cells_within_radius
```

---

**Description**

Calculates the average percentage of cells of a target cell type within a radius from the cells with a reference cell type. The calculation is done per reference cell, so runtime will depend on the number of reference cells present. Output is a single value (the mean for the image).

**Usage**

```
average_percentage_of_cells_within_radius(
  spe_object,
  reference_celltype,
  target_celltype,
  radius = 100,
  feature_colname
)
```

**Arguments**

spe_object	SpatialExperiment object in the form of the output of <a href="#">format_image_to_spe</a> .
reference_celltype	String specifying the cell type of reference cells.
target_celltype	String specifying the cell type for target cells
radius	Integer specifying the radius of search for cells around the reference cells. Radii of ~100 are recommended. If too small, too few cells might be present.
feature_colname	String specifying the column with the desired cell type annotations.

**Value**

A numeric vector and a plot are returned

**Examples**

```
average_percentage_of_cells_within_radius(SPIAT::defined_image, "Tumour",
"Immune3", radius = 100, "Cell.Type")
```

---

```
calculate_cell_proportions
      calculate_cell_proportions
```

---

**Description**

Calculates the number and proportion of each cell type.

**Usage**

```
calculate_cell_proportions(
  spe_object,
  reference_celltypes = NULL,
  celltypes_to_exclude = NULL,
  feature_colname = "Phenotype",
  plot.image = TRUE
)
```

**Arguments**

spe_object	SpatialExperiment object in the form of the output of <a href="#">format_image_to_spe</a> .
reference_celltypes	String Vector specifying reference cell types. If NULL (default), then the proportion of each cell type against all cells is returned. Alternatively, a custom vector of cell types can be used as input, and these will be used as the denominator in the calculation of the proportions.

celltypes_to_exclude	String Vector specifying cell types to exclude. For example "OTHER" will exclude that celltype from the Total. If NULL, all cell types are included.
feature_colname	String. Column of cells to choose the cell type from (e.g. Phenotype, Cell.Type, etc).
plot.image	Boolean. Whether to plot the barplot of the cell percentages. By default is TRUE.

**Value**

A data.frame is returned

**Examples**

```
calculate_cell_proportions(SPIAT::defined_image, reference_celltypes = NULL,
celltypes_to_exclude = "Others", feature_colname="Cell.Type", plot.image = FALSE)
```

---

```
calculate_cross_functions
      calculate_cross_functions
```

---

**Description**

Compute and plot the cross functions between two specified cell types. This function implements the cross functions from [spatstat] package.

**Usage**

```
calculate_cross_functions(
  spe_object,
  method = "Kcross",
  cell_types_of_interest,
  feature_colname,
  plot_results = TRUE,
  dist = NULL
)
```

**Arguments**

spe_object	SpatialExperiment object in the form of the output of <a href="#">format_image_to_spe</a> .
method	String that is the method for dependence calculation. Options: "Gcross", "Kcross", "Kcross.inhom", "Lcross", "Jcross". Default method is "Kcross".
cell_types_of_interest	String Vector. Cell types of interest.
feature_colname	String that is the name of the column of the types.



plot_results	Boolean. TRUE if result to be plotted, FALSE if not. In either case, an object with the results is returned
dist	Number (OPTIONAL) The largest distance between two cell types at which K function is evaluated. If NULL, use the default distances set by cross functions.

**Value**

An object of class "fv" defined in 'spatstat' package.

**Examples**

```
df_cross <- calculate_cross_functions(SPIAT::defined_image,
  method = "Kcross", cell_types_of_interest = c("Tumour", "Immune3"),
  feature_colname = "Cell.Type", dist = 100)
```

---

calculate\_distance\_to\_margin

*calculate the distances of each cell to the margin*

---

**Description**

Returns a SPE object with the minimum distance from cells of interest (CoI) to the identified bordering cells.

**Usage**

```
calculate_distance_to_margin(spe_object)
```

**Arguments**

spe\_object      SpatialExperiment object. It should contain information of the detected bordering cells ('colData()' has 'Region' column).

**Value**

An spe\_object with a 'Distance.To.Border' column is returned.

**Examples**

```
spe_border <- identify_bordering_cells(SPIAT::defined_image,
  reference_cell = "Tumour", feature_colname = "Cell.Type", n_to_exclude = 10)
spe_dist <- calculate_distance_to_margin(spe_border)
```

---

calculate\_entropy      *calculate\_entropy*

---

### Description

If arg 'radius' is not specified, the function returns the entropy of the cell types of interest for the whole image. If arg 'radius' is specified, the function returns a data frame where each row is a reference cell and the columns stores the entropy of the cell types of interest in each circle of the reference cells.

### Usage

```
calculate_entropy(
  spe_object,
  cell_types_of_interest,
  feature_colname = "Phenotype",
  radius = NULL
)
```

### Arguments

**spe\_object**      SpatialExperiment object in the form of the output of [format\\_image\\_to\\_spe](#).

**cell\_types\_of\_interest**      String Vector. Cell types of interest. If arg 'radius' is not NULL, the first cell type is considered as reference cell type. Circles of the specified radius will be drawn around the reference cells and the entropy of cell types will be calculated for each of the reference cells.

**feature\_colname**      String specifying the column the cell types are from.

**radius**      (OPTIONAL) Numeric. The maximum radius around a reference cell for another cell to be considered an interaction.

### Value

A dataframe or a number depending on the argument radius

### Examples

```
calculate_entropy(SPIAT::defined_image,
  cell_types_of_interest = c("Immune1", "Immune2"),
  feature_colname = "Cell.Type")
```

---

```
calculate_minimum_distances_between_celltypes  
    calculate_minimum_distances_between_celltypes
```

---

**Description**

Returns the distance of the closest cell of a specific type from each reference cell.

**Usage**

```
calculate_minimum_distances_between_celltypes(  
  spe_object,  
  feature_colname,  
  cell_types_of_interest = NULL  
)
```

**Arguments**

`spe_object` SpatialExperiment object in the form of the output of [format\\_image\\_to\\_spe](#).  
`feature_colname` String of the feature column of cells to choose the cell types from (e.g. Cell.Type, Cell.Type2, etc).  
`cell_types_of_interest` String Vector of marker combinations to consider is FALSE.

**Value**

A data.frame is returned

**Examples**

```
min_dists <- calculate_minimum_distances_between_celltypes(  
  SPIAT::defined_image, feature_colname = "Cell.Type",  
  cell_types_of_interest = c("Tumour", "Immune1"))
```

---

```
calculate_pairwise_distances_between_celltypes  
    calculate_pairwise_distances_between_celltypes
```

---

**Description**

Returns the pairwise distances between cells of different types. If none of the cell types are found, it will print an error message and return a vector of NAs.

**Usage**

```
calculate_pairwise_distances_between_celltypes(
  spe_object,
  cell_types_of_interest = NULL,
  feature_colname
)
```

**Arguments**

`spe_object` SpatialExperiment object in the form of the output of [format\\_image\\_to\\_spe](#).

`cell_types_of_interest` String Vector containing cell types to be considered, if NULL, all cell type combinations will be calculated.

`feature_colname` String of the name the feature column with the cell types of interest to be considered.

**Value**

A data.frame is returned.

**Examples**

```
calculate_pairwise_distances_between_celltypes(SPIAT::defined_image,
  cell_types_of_interest = c("Tumour", "Immune1"),
  feature_colname = "Cell.Type")
```

---

```
calculate_percentage_of_grids
  calculate_percentage_of_grids
```

---

**Description**

Takes the result of [grid\\_metrics](#) (a RasterLayer object) and calculates the percentage of the grid squares whose values are above or below a specified threshold.

**Usage**

```
calculate_percentage_of_grids(raster_obj, threshold, above)
```

**Arguments**

`raster_obj` Raster object in the form of the output of [grid\\_metrics](#).

`threshold` Numeric. The threshold for defining the pattern.

`above` Boolean. Indicating whether the pattern is above (TRUE) or below (FALSE) the threshold.

**Value**

A number is returned

**Examples**

```
grid <- grid_metrics(SPIAT::defined_image, FUN = calculate_entropy, n_split = 5,
  cell_types_of_interest=c("Tumour", "Immune3"), feature_colname = "Cell.Type")
calculate_percentage_of_grids(grid, threshold = 0.75, above = TRUE)
```

---

```
calculate_proportions_of_cells_in_structure
  calculate_proportions_of_cells_in_structure
```

---

**Description**

Calculate the proportion of cells of interest in each defined tissue structure relative to all cells in each structure and relative to the same cell type in the whole image.

**Usage**

```
calculate_proportions_of_cells_in_structure(
  spe_object,
  cell_types_of_interest,
  feature_colname
)
```

**Arguments**

`spe_object` SpatialExperiment object in the form of the output of [format\\_image\\_to\\_spe](#).  
`cell_types_of_interest` String Vector of cell types to consider.  
`feature_colname` String. The name of the column where the cell types of interest are under.

**Value**

A data.frame

**Examples**

```
spe_border <- identify_bordering_cells(SPIAT::defined_image,
  reference_cell = "Tumour", feature_colname = "Cell.Type", n_to_exclude = 10)
spe_dist <- calculate_distance_to_margin(spe_border)
spe_structure <- define_structure(spe_dist,
  cell_types_of_interest = c("Immune1", "Immune2", "Immune3"),
  feature_colname = "Cell.Type", n_margin_layers = 5)
calculate_proportions_of_cells_in_structure(spe_structure,
  cell_types_of_interest = c("Immune1", "Immune3"), feature_colname="Cell.Type")
```

---

```
calculate_spatial_autocorrelation
      calculate_spatial_autocorrelation
```

---

**Description**

Takes the result of `grid_metrics` (a RasterLayer object) and calculate its spatial autocorrelation.

**Usage**

```
calculate_spatial_autocorrelation(raster_obj, metric = "globalmoran", d = NULL)
```

**Arguments**

<code>raster_obj</code>	Raster object in the form of the output of <code>grid_metrics</code> .
<code>metric</code>	String. The method for calculating spatial autocorrelation. Choose from "globalmoran" and "GearyC".
<code>d</code>	Numeric. Upper bound local distance. The argument 'd2' from function <code>moran</code> . Default is NULL and the distance will be calculated automatically from the number of splits and the extent of the grid image.

**Value**

A number is returned

**Examples**

```
grid <- grid_metrics(SPIAT::defined_image, FUN = calculate_entropy,
  n_split = 5, cell_types_of_interest=c("Tumour","Immune3"),
  feature_colname = "Cell.Type")
calculate_spatial_autocorrelation(grid, metric = "globalmoran")
```

---

```
calculate_summary_distances_between_celltypes
      calculate_summary_distances_between_celltypes
```

---

**Description**

Returns the mean, median and standard deviation of the minimum/pairwise distances between phenotypes.

**Usage**

```
calculate_summary_distances_between_celltypes(df)
```

**Arguments**

`df` Data.frame containing the distance output between cell types. The functions that generate the distances can be [calculate\\_minimum\\_distances\\_between\\_celltypes](#) and [calculate\\_pairwise\\_distances\\_between\\_celltypes](#).

**Value**

A data frame is returned

**Examples**

```
# for pairwise dist
pairwise_dist <- calculate_pairwise_distances_between_celltypes(
  SPIAT::defined_image, cell_types_of_interest = c("Tumour", "Immune1"),
  feature_colname = "Cell.Type")
summary_distances <- calculate_summary_distances_between_celltypes(pairwise_dist)

# for minimum dist
min_dists <- calculate_minimum_distances_between_celltypes(
  SPIAT::defined_image, cell_types_of_interest = c("Tumour", "Immune1"),
  feature_colname = "Cell.Type")
summary_distances <- calculate_summary_distances_between_celltypes(min_dists)
```

---

```
calculate_summary_distances_of_cells_to_borders
  calculate_summary_distances_of_cells_to_borders
```

---

**Description**

Returns the mean, median and standard deviation of the distances between a specified cell type to the border.

**Usage**

```
calculate_summary_distances_of_cells_to_borders(
  spe_object,
  cell_types_of_interest,
  feature_colname = "Cell.Type"
)
```

**Arguments**

`spe_object` SpatialExperiment object. It should contain information of tissue structure and cell distances to the tissue region border (`colData()` has `'Region'` and `'Distance.To.Border'` columns).

`cell_types_of_interest` String Vector of cell types to consider.

`feature_colname` String specifying which column the interested cell types are from.

**Value**

A data.frame is returned

**Examples**

```
spe_border <- identify_bordering_cells(SPIAT::defined_image,
reference_cell = "Tumour", feature_colname = "Cell.Type", n_to_exclude = 10)
spe_dist <- calculate_distance_to_margin(spe_border)
spe_structure <- define_structure(spe_dist, cell_types_of_interest =
c("Immune1", "Immune2", "Immune3"), feature_colname = "Cell.Type",
n_margin_layers = 5)
calculate_summary_distances_of_cells_to_borders(spe_structure,
cell_types_of_interest = c("Immune1", "Immune3"), feature_colname = "Cell.Type")
```

---

composition\_of\_neighborhoods

*composition\_of\_neighborhoods*

---

**Description**

Returns a data.frame which contains the percentages of cells with a specific marker within each neighborhood. and the number of cells in the neighborhood.

**Usage**

```
composition_of_neighborhoods(spe_object, feature_colname)
```

**Arguments**

spe\_object      SpatialExperiment that is the output of `identify_neighborhoods`.  
feature\_colname      String. Column with cell types.

**Value**

A data.frame is returned

**Examples**

```
neighborhoods <- identify_neighborhoods(image_no_markers,
method = "hierarchical", min_neighborhood_size = 100,
cell_types_of_interest = c("Immune", "Immune1", "Immune2"), radius = 50,
feature_colname = "Cell.Type")
neighborhoods_vis <- composition_of_neighborhoods(neighborhoods,
feature_colname="Cell.Type")
```



---

compute\_gradient      *compute\_gradient*

---

### Description

The function sweeps over circles of a range of radii surrounding reference cells and calculates the metrics at the radii. Metrics used with function need two conditions: 1) have a 'radius' parameter. 2) return a single number. For metrics that do not return a single number, users can wrap them in a new function that returns a number and then pass the new function to 'compute\_gradient()'.

### Usage

```
compute_gradient(spe_object, radii, FUN, ...)
```

### Arguments

spe_object	SpatialExperiment object in the form of the output of <a href="#">format_image_to_spe</a> .
radii	Numeric Vector specifying the range of radii for the metrics to be calculated.
FUN	Variable name specifying the metric.
...	Arguments of FUN

### Value

A list of the metrics under all radii

### Examples

```
gradient_positions <- c(30, 50, 100)
gradient_entropy <- compute_gradient(SPIAT::defined_image,
  radii = gradient_positions, FUN = calculate_entropy,
  cell_types_of_interest = c("Immune1", "Immune2"),
  feature_colname = "Cell.Type")
```

---

crossing\_of\_crossK      *crossing\_of\_crossK*

---

### Description

Determine if there is a crossing in the cross K curves, to further detect the existence of potential immune rings.

### Usage

```
crossing_of_crossK(df.cross)
```

**Arguments**

`df.cross` Data.frame. The output of `calculate_cross_functions`. Containing the positions of the two curves. Columns contain "r", "border" and "theo".

**Value**

A number. The percentage of the crossing position of the specified distance.

**Examples**

```
df_cross <- calculate_cross_functions(SPIAT::defined_image, method="Kcross",
  cell_types_of_interest = c("Tumour","Immune3"),
  feature_colname = "Cell.Type", dist = 100)
crossing_of_crossK(df_cross)
```

---

<code>defined_image</code>	<i>SPE object of a simulated image with defined cell types based on marker combinations.</i>
----------------------------	--

---

**Description**

A dataset that contains a formatted spe object with cell ids, phenotypes, defined cell types in 'colData()' and marker intensities in 'assays()'. (The cell locations are the same with the cells in `simulated_image`).

**Usage**

```
defined_image
```

**Format**

An spe object. Assay contains 5 rows (markers) and 4951 columns (cells); colData contains 4951 rows (cells) and 3 columns (features).

**See Also**

`simulated_image` `image_no_markers`

---

define_celltypes	<i>define_celltypes</i>
------------------	-------------------------

---

### Description

Define new cell types based on the existing cell types (categories) under a selected column (e.g. base on marker combinations under "Phenotype" column). This function will create a new column to store the new cell types.

### Usage

```
define_celltypes(
  spe_object,
  categories = NULL,
  category_colname = "Phenotype",
  names = NULL,
  new_colname = "Cell.Type",
  print_names = FALSE
)
```

### Arguments

spe_object	SpatialExperiment object in the form of the output of <a href="#">format_image_to_spe</a> .
categories	Vector. Names of the old cell types to be defined; if NULL, the function will use predefined categories and names
category_colname	(Phenotype) String specifying the name of the column having the categories to be defined, by default "Phenotype".
names	Vector of new names assigned to the selected categories; if NULL, the function will use predefined categories and names. Should be of the same length of 'categories'.
new_colname	(Optional) String specifying the name of the column to be added, by default "Cell.Type".
print_names	(Optional) Boolean if the user wants the original and new names printed. Default is FALSE.

### Details

Users need to specify the names of the old cell categories and under which column the old cell categories exist. Then the users specify the names of the new cell types and the name of the new column to store the new cell types. Any cell categories that are not specified in 'categories' arg but present in the image will be defined as "Undefined" in the new column.

### Value

An new SPE object is returned

**Examples**

```
# the selected column is:
category_colname = "Phenotype"
# define the following marker combinations:
categories <- c("Tumour_marker", "Immune_marker1,Immune_marker2",
"Immune_marker1,Immune_marker3",
"Immune_marker1,Immune_marker2,Immune_marker4", "OTHER")
# the new defined cell names:
names = c("Tumour", "Immune1", "Immune2","Immune3", "Others")
# the new names are stored under this column:
new_colname <- "Cell.Type"

defined_spe <- define_celltypes(SPIAT::simulated_image,
categories = categories, category_colname = category_colname, names = names,
new_colname = new_colname)
```

---

define_structure	<i>define_structure</i>
------------------	-------------------------

---

**Description**

After identifying the bordering cells of tissue regions and calculating the distances of each cell to the bordering cells, this function further identifies the cells that are located in the inside and outside of the identified regions, and in the internal and external margins. It also identifies particular types of cells that are infiltrated, stromal, internal margin or external margin cells.

**Usage**

```
define_structure(
  spe_object,
  cell_types_of_interest,
  feature_colname = "Cell.Type",
  n_margin_layers = 5,
  margin_dist = NULL
)
```

**Arguments**

spe_object	SpatialExperiment object that contains information of tumour bordering cells and cell distances to border ('colData()' has 'Region' and 'Distance.To.Border' columns).
cell_types_of_interest	String Vector of the names of the particular types of cells.
feature_colname	String Specifying the column that contains the names of the immune cells.
n_margin_layers	Integer. The number of layers of cells that compose the internal/external margins. Default is 5.

`margin_dist` Numeric. The width of the internal/external margins. Default is NULL. Only use when `'n_margin_layers'` is NULL.

### Value

A new `spe` object is returned. Under the `'Region'` column, there will be potential categories including `'Border'` - the bordering cells, `'Infiltrated.CoI'` - cells of interest that present inside of the tissue regions, `'Inside'` - cells within the regiona excluding the `'Infiltrated.CoI'` cells and the cells at internal margin, `'Stromal.CoI'` - cells of interest that present outside of the tissue regions, `'Outside'` - cells outside of the tissue regions excluding the `'Stromal.CoI'` cells, `'Internal.margin.CoI'` - cells of interest that are in the internal margin of the tissue regions, `'Internal.margin'` - cells in the internal margin of the tissue regions excluding the `'Internal.margin.CoI'` cells, `'External.margin.CoI'` - cells of interest that are in the external margin of the tissue regions, `'External.margin'` - cells in the external margin of the tissue regions excluding the `'External.margin.CoI'` cells.

### Examples

```
spe_border <- identify_bordering_cells(SPIAT::defined_image,
reference_cell = "Tumour", feature_colname = "Cell.Type", n_to_exclude = 10)
spe_dist <- calculate_distance_to_margin(spe_border)
spe_structure <- define_structure(spe_dist,
cell_types_of_interest = c("Immune1", "Immune2", "Immune3"),
feature_colname = "Cell.Type", n_margin_layers = 5)
plot_cell_categories(spe_structure, feature_colname = "Structure")
```

---

dimensionality\_reduction\_plot  
*Dimensionality reduction plot*

---

### Description

Generates the dimensionality reduction plots (UMAP or tSNE) based on marker intensities. Cells are grouped by the categories under the selected column.

### Usage

```
dimensionality_reduction_plot(
  spe_object,
  plot_type = "UMAP",
  scale = TRUE,
  perplexity = 30,
  feature_colname
)
```

**Arguments**

spe_object	SpatialExperiment object in the form of the output of <a href="#">format_image_to_spe</a> .
plot_type	String. Choose from "UMAP" and "TSNE".
scale	Boolean. Whether scale the marker intensities.
perplexity	Numeric. Perplexity parameter of the Rtsne function (should be positive and no bigger than $3 * \text{perplexity} < n - 1$ , where n is the number of cells).
feature_colname	String. Specify the column name to group the cells.

**Value**

A plot

**Examples**

```
dimensionality_reduction_plot(SPIAT::simulated_image, plot_type = "TSNE",
feature_colname = "Phenotype")
```

---

entropy\_gradient\_aggregated

*The aggregated gradient of entropy and the peak of the gradient*

---

**Description**

This function first calculates the entropy within circles of each reference cell at each radius. Then at each radius, the entropy of all circles surrounding each cell are aggregated into one number. The function sweeps over the specified radii and calculates the aggregated entropy under each radius.

**Usage**

```
entropy_gradient_aggregated(
  spe_object,
  cell_types_of_interest,
  feature_colname,
  radii
)
```

**Arguments**

spe_object	SpatialExperiment object in the form of the output of <a href="#">format_image_to_spe</a> .
cell_types_of_interest	String Vector. The cell types that the entropy is computed on.
feature_colname	String. The column name of the interested cell types.
radii	Numeric Vector. A vector of radii within a circle of a reference cell where the entropy is computed on.

**Value**

A list of the gradient of entropy and the peak

**Examples**

```
gradient_pos <- seq(50, 500, 50)
gradient_results <- entropy_gradient_aggregated(SPIAT::defined_image,
cell_types_of_interest = c("Tumour", "Immune3"),
feature_colname = "Cell.Type", radii = gradient_pos)
plot(1:10, gradient_results$gradient_df[1, 3:12])
```

---

format\_cellprofiler\_to\_spe

*Format a cellprofiler image into a SpatialExperiment object*

---

**Description**

Reads in spatial data in the form of cell coordinates, cell phenotypes (if available), and marker intensities and transforms to a SpatialExperiment object. The assay stores the intensity level of every marker (rows) for every cell (columns). Cell phenotype is stored under 'colData()'. Cell x and y coordinates are stored under 'spatialCoords()'. Note that if the data does not include these parameters, we recommend adding it to the output from cellprofiler with NAs in columns.

**Usage**

```
format_cellprofiler_to_spe(
  path = NULL,
  markers = NULL,
  intensity_columns_interest = NULL
)
```

**Arguments**

path	String of the path location cellprofiler csv file.
markers	String Vector containing the markers used for staining.
intensity_columns_interest	String Vector with the names of the columns with the level of each marker. Column names must match the order of the 'markers' parameter.

**Details**

Note when specifying 'markers', please use "DAPI" to replace "DNA" due to implementation. The output data will include "DAPI" instead of "DNA".

**Value**

A SpatialExperiment object is returned

**Examples**

```
path <- system.file("extdata", "tiny_cellprofiler.txt.gz", package = "SPIAT")
markers <- c("Marker1", "Marker2", "Marker3", "Marker4", "Marker5", "DAPI",
"Marker6")
intensity_columns_interest <- c("Intensity_MeanIntensity_Marker1_rs",
"Intensity_MeanIntensity_Marker2_rs", "Intensity_MeanIntensity_Marker3_rs",
"Intensity_MeanIntensity_Marker4_rs", "Intensity_MeanIntensity_Marker5_rs",
"Intensity_MeanIntensity_DAPI_rs", "Intensity_MeanIntensity_Marker6_rs")
formatted_cellprofiler <- format_cellprofiler_to_spe(path = path,
markers = markers, intensity_columns_interest = intensity_columns_interest)
```

---

format\_codex\_to\_spe     *Format a CODEX image into a SpatialExperiment object*

---

**Description**

Reads in spatial data in the form of cell coordinates, cell phenotypes (if available), and marker intensities and transforms to a ‘SpatialExperiment’ object. The assay stores the intensity level of every marker (rows) for every cell (columns). Cell phenotype is stored under colData. Cell x and y coordinates are stored under ‘spatialCoords()’ field.

**Usage**

```
format_codex_to_spe(path = NULL, markers, path_to_codex_cell_phenotypes = NULL)
```

**Arguments**

path                    String of the path location of CODEX csv file.  
markers                 String Vector containing the markers used for staining.  
path\_to\_codex\_cell\_phenotypes  
                         String of the path to the Cluster ID/Cell type file.

**Value**

A SpatialExperiment object is returned

**Examples**

```
path <- system.file("extdata", "tiny_codex.csv.gz", package = "SPIAT")
path_to_codex_cell_phenotypes <- system.file("extdata",
"tiny_codex_phenotypes.txt.gz", package = "SPIAT")
markers <- c("CD45", "Ly6C", "CD27", "CD5", "CD79b")
formatted_codex <- format_codex_to_spe(path = path, markers = markers,
path_to_codex_cell_phenotypes = path_to_codex_cell_phenotypes)
```



---

format\_colData\_to\_spe *format\_colData\_to\_spe*

---

### Description

Format a data frame into a SpatialExperiment class where the count assay is empty every cell (columns), cell phenotypes are stored under colData() and cell coordinates are stored under spatialCoords().

### Usage

```
format_colData_to_spe(df)
```

### Arguments

df                      Data frame that contains cell coordinates, phenotypes (if available) and other cell properties. The rownames should be cell ID

### Value

An SpatialExperiment object

### Examples

```
df <- data.frame(row.names = c("Cell_1", "Cell_2"), Cell.X.Position = c(2,5),  
Cell.Y.Position = c(3.3, 8), Phenotypes = c("CD3", "CD3,CD8"))  
spe <- format_colData_to_spe(df)
```

---

format\_halo\_to\_spe      *Format a HALO image into a SpatialExperiment object*

---

### Description

Reads in HALO data in the form of cell coordinates, cell phenotypes (if available), and marker intensities and transforms to a 'SpatialExperiment' object. The assay stores the intensity level of every marker (rows) for every cell (columns). Cell x and y coordinates are stored under 'spatialCoords()'. Cell phenotype and other cell properties are stored under colData. The cell properties to be included are Cell.Area, Nucleus.Area and Cytoplasm.Area. Note that if the data does not include these parameters, we recommend adding it to the output from HALO with NAs in columns.

**Usage**

```
format_halo_to_spe(
  path = NULL,
  markers = NULL,
  locations = NULL,
  dye_columns_interest = NULL,
  intensity_columns_interest = NULL
)
```

**Arguments**

**path** String of the path location of HALO text file.

**markers** String Vector containing the markers used for staining.

**locations** (Optional) Vector containing the locations of markers used for staining. Location can be either "Nucleus", "Cytoplasm" or "Membrane". This is used to select the Intensity column and can be used instead of 'intensity\_columns\_interest'.

**dye\_columns\_interest** (Optional) Use if locations is not specified. Vector of names of the columns with the marker status (i.e. those indicating 1 or 0 for whether the cell is positive or negative for the marker). Column names must match the order of the 'markers' parameter.

**intensity\_columns\_interest** (Optional) Use if locations is not specified. Vector with the names of the columns with the level of each marker. Column names must match the order of the 'markers' parameter.

**Value**

A SpatialExperiment object is returned

**Examples**

```
raw_halo_data <- system.file("extdata", "tiny_halo.csv.gz", package="SPIAT")
markers <- c("DAPI", "CD3", "PDL-1", "CD4", "CD8", "AMACR")
intensity_columns_interest <- c("Dye 1 Nucleus Intensity",
"Dye 2 Cytoplasm Intensity", "Dye 3 Membrane Intensity",
"Dye 4 Cytoplasm Intensity", "Dye 5 Cytoplasm Intensity",
"Dye 6 Cytoplasm Intensity")
dye_columns_interest <-c("Dye 1 Positive Nucleus", "Dye 2 Positive Cytoplasm",
"Dye 3 Positive Membrane", "Dye 4 Positive Cytoplasm",
"Dye 5 Positive Cytoplasm", "Dye 6 Positive Cytoplasm")
formatted_HALO <- format_halo_to_spe(path=raw_halo_data, markers=markers,
intensity_columns_interest=intensity_columns_interest,
dye_columns_interest=dye_columns_interest)
```

---

format\_image\_to\_spe     *Format an image into a SpatialExperiment object*

---

## Description

Reads in spatial data in the form of cell coordinates, cell phenotypes (if available), and marker intensities and transforms to a SpatialExperiment object. The assay stores the intensity level of every marker (rows) for every cell (columns). Cell phenotype is stored under 'colData()'. Cell x and y coordinates are stored under 'spatialCoords()' field. The function can read in data in general format (manually constructed input), or data from other platforms including inForm, HALO, CODEX and cellprofiler. Alternatively, users can use the specific function for each format.

## Usage

```
format_image_to_spe(
  format = "general",
  intensity_matrix = NULL,
  phenotypes = NULL,
  coord_x = NULL,
  coord_y = NULL,
  path = NULL,
  markers = NULL,
  locations = NULL,
  intensity_columns_interest = NULL,
  dye_columns_interest = NULL,
  path_to_codex_cell_phenotypes = NULL
)
```

## Arguments

format	String specifying the format of the data source. Default is "general" (RECOMMENDED), where the cell phenotypes, coordinates and marker intensities are imported manually by the user. Other formats include "inForm", "HALO", "cellprofiler" and "CODEX".
intensity_matrix	(Optional) For "general" format. A matrix of marker intensities or gene expression where the column names are the Cell IDs, and the rownames the marker.
phenotypes	(Optional) For "general" format. String Vector of cell phenotypes in the same order in which they appear in 'intensity_matrix'. If no phenotypes available, then a vector of NAs can be used as input. Note that the combination of markers (e.g. CD3,CD4) needs to be used instead of the cell type name (e.g. helper T cells).
coord_x	(Optional) For "general" format. Numeric Vector with the X coordinates of the cells. The cells must be in the same order as in the 'intensity_matrix'.
coord_y	(Optional) For "general" format. Numeric Vector with the Y coordinates of the cells. The cells must be in the same order as in the 'intensity_matrix'.

path	(Optional) For formats other than "general". String of the path location of the source file.
markers	For formats other than "general". String Vector containing the markers used for staining. These must be in the same order as the marker columns in the input file, and must match the marker names used in the input file. One of the markers must be "DAPI".
locations	(Optional) For "inForm" and "HALO". String Vector containing the locations of markers used for staining. Location can be either "Nucleus", "Cytoplasm" or "Membrane". This is used to select the Intensity column and can be used instead of 'intensity_columns_interest'.
intensity_columns_interest	(Optional) For "inForm" and "HALO", use if 'locations' is not specified. For "cellprofiler", mandatory. Vector with the names of the columns with the level of each marker. Column names must match the order of the 'markers' parameter.
dye_columns_interest	(Optional) For "HALO". Use if locations is not specified. Vector of names of the columns with the marker status (i.e. those indicating 1 or 0 for whether the cell is positive or negative for the marker). Column names must match the order of the 'markers' parameter.
path_to_codex_cell_phenotypes	(Optional) For "CODEX".String of the path to the Cluster ID/Cell type file.

## Details

If the user inputs 'intensity\_matrix', please make sure the 'colnames' of the intensity matrix are the cell IDs. If the 'intensity\_matrix' is 'NULL', the function will automatically assign IDs to the cells. Note for "cellprofiler" format, when specifying 'markers', please use "DAPI" to replace "DNA" due to implementation. The output data will include "DAPI" instead of "DNA".

The format of "Phenotype" column: For example, a cell positive for both "CD3" and "CD4" markers has the "CD3,CD4" **\*\*cell phenotype\*\***. The phenotype has to be strictly formatted in such way where each positive marker has to be separated by a coma, with no space in between, and the order of the positive markers has to be the same as the order in the assay.

## Value

A SpatialExperiment object is returned

## See Also

[format\\_inform\\_to\\_spe](#) [format\\_halo\\_to\\_spe](#) [format\\_codex\\_to\\_spe](#) [format\\_cellprofiler\\_to\\_spe](#)

## Examples

```
#Construct a marker intensity matrix (rows are markers, columns are cells)
intensity_matrix <- matrix(c(14.557, 0.169, 1.655, 0.054, 17.588, 0.229,
1.188, 2.074, 21.262, 4.206, 5.924, 0.021), nrow = 4, ncol = 3)
# define marker names as rownames
rownames(intensity_matrix) <- c("DAPI", "CD3", "CD4", "AMACR")
```

```

# define cell IDs as colnames
colnames(intensity_matrix) <- c("Cell_1", "Cell_2", "Cell_3")
# Construct a dummy metadata (phenotypes, x/y coordinates)
# the order of the elements in these vectors correspond to the cell order
# in `intensity_matrix`
phenotypes <- c("OTHER", "AMACR", "CD3,CD4")
coord_x <- c(82, 171, 184)
coord_y <- c(30, 22, 38)

formatted_image <- format_image_to_spe(intensity_matrix=intensity_matrix,
phenotypes = phenotypes, coord_x = coord_x, coord_y = coord_y)

```

---

format\_inform\_to\_spe *Format an inForm image into a SpatialExperiment object*

---

## Description

Reads in inForm data in the form of cell coordinates, cell phenotypes (if available), and marker intensities and transforms to a SpatialExperiment object. The assay stores the intensity level of every marker (rows) for every cell (columns). Cell phenotype, x and y coordinates and other cell properties are stored under colData. The cell properties to include are Cell.Area, Nucleus.Area, Nucleus.Compactness, Nucleus.Axis.Ratio, and Cell.Axis.Ratio. Note that if the data does not include these parameters, we recommend adding it to the output from inForm with NAs in columns.

## Usage

```

format_inform_to_spe(
  path,
  markers,
  locations = NULL,
  intensity_columns_interest = NULL
)

```

## Arguments

path	String of the path location of inForm text file.
markers	String Vector containing the markers used for staining.
locations	(Optional) String Vector containing the locations of markers used for staining. Location can be either "Nucleus", "Cytoplasm" or "Membrane". This is used to select the Intensity column and can be used instead of 'intensity_columns_interest'.
intensity_columns_interest	(Optional) Use if 'locations' is not specified. Vector with the names of the columns with the level of each marker. Column names must match the order of the 'markers' parameter.

## Value

A SpatialExperiment object is returned

**Examples**

```
raw_inform_data<-system.file("extdata","tiny_inform.txt.gz",package="SPIAT")
markers <- c("DAPI", "CD3", "PD-L1", "CD4", "CD8", "AMACR")
locations <- c("Nucleus", "Cytoplasm", "Membrane", "Cytoplasm", "Cytoplasm",
"Cytoplasm")
formatted_inform <- format_inform_to_spe(path=raw_inform_data,
markers=markers, locations=locations)
```

---

format\_spe\_to\_ppp      *Format SPE object as a ppp object ('spatstat' package)*

---

**Description**

Formats an spe object into a ppp object which has the x,y coordinates, phenotypes as markers window specifies the range of x and y coordinates

**Usage**

```
format_spe_to_ppp(
  spe_object,
  window_pol = FALSE,
  feature_colname = "Phenotype"
)
```

**Arguments**

spe\_object      SpatialExperiment object in the form of the output of format\_image\_to\_spe.  
window\_pol      Optional Boolean Specifying if the window is polygon.  
feature\_colname      String specifying the feature column of interest.

**Value**

A ppp object is returned (defined in 'spatstat' package)

**Examples**

```
ppp_object<-format_spe_to_ppp(SPIAT::defined_image,
feature_colname = "Cell.Type")
```

---

grid_metrics	<i>Split an image into grid and calculates a metric for each grid square</i>
--------------	--

---

**Description**

Calculates a specified metric for each grid tile in the image and plots the metrics for the grid tiles.

**Usage**

```
grid_metrics(spe_object, FUN, n_split, ...)
```

**Arguments**

spe_object	SpatialExperiment object in the form of the output of <a href="#">format_image_to_spe</a> .
FUN	Variable name specifying the metric to be calculated.
n_split	Integer specifying the number of splits for the calculation of metrics. This number is the splits on each side (e.g. 'n_split' = 3 means the image will be split into 9 tiles.)
...	Arguments of FUN

**Value**

A list of the metrics of all grid tiles

**Examples**

```
grid <- grid_metrics(SPIAT::defined_image, FUN = calculate_entropy, n_split = 5,
  cell_types_of_interest=c("Tumour","Immune3"), feature_colname = "Cell.Type")
```

---

identify_bordering_cells	<i>identify_bordering_cells</i>
--------------------------	---------------------------------

---

**Description**

Identify the cells bordering a group of cells of a particular phenotype, and calculate the number of clustered groups of this cell type.

**Usage**

```
identify_bordering_cells(
  spe_object,
  reference_cell,
  feature_colname = "Cell.Type",
  ahull_alpha = NULL,
  n_to_exclude = 10,
  plot_final_border = TRUE
)
```

**Arguments**

spe_object	SpatialExperiment object in the form of the output of <code>format_image_to_spe</code> .
reference_cell	String. Cells of this cell type will be used for border detection.
feature_colname	String that specifies the column of 'reference_cell'.
ahull_alpha	Number specifying the parameter for the alpha hull algorithm. The larger the number, the more cells will be included in one cell cluster.
n_to_exclude	Integer. Clusters with cell count under this number will be deleted.
plot_final_border	Boolean if plot the identified bordering cells.

**Details**

The bordering cell detection algorithm is based on computing an alpha hull (Hemmer et al., 2020), a generalization of convex hull (Green and Silverman, 1979). The cells detected to be on the alpha hull are identified as the bordering cells.

**Value**

A new SPE object is returned. The SPE object has a 'Region' column with "Border", "Inside" and "Outside" categories. The returned object also has an attribute saving the number of clusters.

**Examples**

```
spe_border <- identify_bordering_cells(SPIAT::defined_image,
reference_cell = "Tumour", feature_colname = "Cell.Type", n_to_exclude = 10)
n_clusters <- attr(spe_border, "n_of_clusters") # get the number of clusters
n_clusters
```

---

identify\_neighborhoods

*identify\_neighborhoods*

---

**Description**

Uses Euclidean distances to identify neighborhoods of cells. Three clustering methods are available, including hierarchical clustering, dbscan, and (Rphenograph).

**Usage**

```
identify_neighborhoods(
  spe_object,
  method = "hierarchical",
  cell_types_of_interest,
  radius,
  min_neighborhood_size = 10,
```



```

    k = 100,
    feature_colname,
    no_pheno = NULL
  )

```

### Arguments

spe_object	SpatialExperiment object in the form of the output of <code>format_image_to_spe</code> .
method	String. The clustering method. Choose from "hierarchical", "dbscan" and "Rphenograph". (Note Rphenograph function is not available for this version yet).
cell_types_of_interest	String Vector of phenotypes to consider.
radius	Numeric specifying the radius of search. Need to specify when 'method' is "hierarchical" or "dbscan".
min_neighborhood_size	Numeric. The minimum number of cells within each cluster. Need to specify when 'method' is "hierarchical" or "dbscan".
k	Numeric. The parameter for "Rphenograph" method.
feature_colname	String. Column from which the cell types are selected.
no_pheno	Cell type corresponding to cells without a known phenotype (e.g. "None", "Other")

### Value

An spe object and a plot is returned. The spe object contains information of the defined neighborhood under "Neighborhood" column. The cells of interest that do not form clusters are labelled "Free\_cell", cells not of interest are labelled 'NA'.

### Examples

```

neighborhoods <- identify_neighborhoods(image_no_markers, method = "hierarchical",
min_neighborhood_size = 100, cell_types_of_interest = c("Immune", "Immune1", "Immune2"),
radius = 50, feature_colname = "Cell.Type")

```

---

image_no_markers	<i>SPE object of a formatted image without marker intensities (simulated by 'spaSim' package)</i>
------------------	---

---

### Description

A dataset that contains a formatted spe object with cell ids and cell types in 'colData()' and cell coordinates in 'spatialCoords()'. This dataset does not contain assays (marker intensities).

### Usage

```
image_no_markers
```

**Format**

An spe object. colData contains 4951 rows (cells) and 3 columns (features).

**See Also**

[defined\\_image](#) [simulated\\_image](#)

---

image_splitter	<i>Split a large image into sub images</i>
----------------	--

---

**Description**

Takes in an image in SpatialExperiment format, splits the image into specified sections and returns a list of SpatialExperiment objects. Users can choose to plot the cell positions in each sub image. Note that this function does not split the assay.

**Usage**

```
image_splitter(
  spe_object,
  number_of_splits,
  plot = FALSE,
  cut_labels = TRUE,
  colour_vector = NULL,
  minX = NULL,
  maxX = NULL,
  minY = NULL,
  maxY = NULL,
  feature_colname = "Phenotype"
)
```

**Arguments**

spe_object	'SpatialExperiment' object in the form of the output of <a href="#">format_image_to_spe</a> .
number_of_splits	Numeric. specifying the number of segments (e.g. 2 = 2x2, 3 = 3x3).
plot	Boolean. Specifies whether the splitted images should be printed in a pdf.
cut_labels	Boolean. Specifies whether to plot where the image had been segmented.
colour_vector	String Vector. If specified, the colours will be used for plotting. If NULL, colors will be generated automatically.
minX	Integer used to specify the minimum x boundary to be splitted.
maxX	Integer used to specify the maximum x boundary to be splitted.
minY	Integer used to specify the minimum y boundary to be splitted.
maxY	Integer used to specify the maximum y boundary to be splitted.
feature_colname	String specifying which column the colouring should be based on. Specify when 'plot' is TRUE. Default is "Phenotype".

**Value**

A list of spe objects is returned. Each data frame represents an image without assay data.

**Examples**

```
split_image <- image_splitter(SPIAT::simulated_image, number_of_splits=3,  
plot = FALSE)
```

---

```
marker_intensity_boxplot  
      marker_intensity_boxplot
```

---

**Description**

Produces boxplots of marker levels for cells phenotyped as being positive for the marker, and those that were phenotyped as being negative.

**Usage**

```
marker_intensity_boxplot(spe_object, marker)
```

**Arguments**

spe_object	SpatialExperiment object in the form of the output of <a href="#">format_image_to_spe</a> .
marker	String. Marker being queried.

**Value**

A plot is returned

**Examples**

```
marker_intensity_boxplot(SPIAT::simulated_image, "Immune_marker1")
```

---

```
marker_prediction_plot
      marker_prediction_plot
```

---

**Description**

Takes in the returned dataframe from `marker_threshold_plot` and generates a .pdf file containing scatter plots of actual intensity and predicted intensity for every marker.

**Usage**

```
marker_prediction_plot(predicted_data, marker)
```

**Arguments**

`predicted_data` Output from `predict_phenotypes`.

`marker` String. Marker to plot

**Value**

A plot is returned

**Examples**

```
predicted_result <- predict_phenotypes(spe_object = simulated_image, thresholds = NULL,
tumour_marker = "Tumour_marker",baseline_markers = c("Immune_marker1", "Immune_marker2",
"Immune_marker3", "Immune_marker4"), reference_phenotypes = TRUE)
marker_prediction_plot(predicted_result, marker = "Tumour_marker")
```

---

```
marker_surface_plot   marker_surface_plot
```

---

**Description**

Generates a 3D surface plot of the level of the selected marker. Note that the image is blurred based on the 'num\_splits' parameter.

**Usage**

```
marker_surface_plot(
  spe_object,
  num_splits,
  marker,
  x_position_min = NULL,
  x_position_max = NULL,
  y_position_min = NULL,
  y_position_max = NULL
)
```

**Arguments**

spe_object	SpatialExperiment object in the form of the output of <a href="#">format_image_to_spe</a> .
num_splits	Integer specifying the number of splits on the image, higher splits equal to higher resolution. Recommendation: 10-100
marker	Marker to plot
x_position_min	Integer specifying the minimum x boundary to be splitted
x_position_max	Integer specifying the maximum x boundary to be splitted
y_position_min	Integer specifying the minimum y boundary to be splitted
y_position_max	Integer specifying the maximum y boundary to be splitted

**Value**

A plot is returned

**Examples**

```
marker_surface_plot(SPIAT::simulated_image, num_splits=15, marker="Immune_marker1")
```

---

```
marker_surface_plot_stack  
  marker_surface_plot_stack
```

---

**Description**

Generates stacked 3D surface plots showing normalized intensity level of specified markers.

**Usage**

```
marker_surface_plot_stack(  
  spe_object,  
  num_splits,  
  markers_to_plot,  
  sep = 1,  
  x_position_min = NULL,  
  x_position_max = NULL,  
  y_position_min = NULL,  
  y_position_max = NULL  
)
```

**Arguments**

spe_object	SpatialExperiment object in the form of the output of <a href="#">format_image_to_spe</a> .
num_splits	Integer specifying the number of splits on the image, higher splits equal to higher resolution. Recommendation: 10-100.
markers_to_plot	Vector of marker names for plotting.
sep	Integer specifying the distance separation between each surface plot. We recommend values in the 1-2 range.
x_position_min	Integer specifying the minimum x boundary to be splitted.
x_position_max	Integer specifying the maximum x boundary to be splitted.
y_position_min	Integer specifying the minimum y boundary to be splitted.
y_position_max	Integer specifying the maximum y boundary to be splitted.

**Value**

A plot is returned

**Examples**

```
marker_surface_plot_stack(SPIAT::simulated_image, num_splits=15,
markers=c("Tumour_marker", "Immune_marker4"))
```

---

```
measure_association_to_cell_properties
      measure_association_to_cell_properties
```

---

**Description**

Plots the density or boxplot of a property of two cell celltypes or compares using t test/wilcoxon rank sum test.

**Usage**

```
measure_association_to_cell_properties(
  spe_object,
  property = "Cell.Area",
  celltypes,
  feature_colname = "Cell.Type",
  method = "density",
  Nucleus.Ratio = FALSE,
  log.scale = FALSE
)
```

**Arguments**

spe_object	SpatialExperiment object in the form of the output of <code>format_image_to_spe</code> .
property	String that is the name of the column of interest.
celltypes	String Vector of celltypes of interest.
feature_colname	String that specifies the column of the cell types.
method	String. The analysis to perform on the selected cell types and property. Options are "density", "box", "t", "wilcox".
Nucleus.Ratio	Boolean whether the ratio of the nucleus size is of interest.
log.scale	Boolean whether to log the data.

**Value**

With method "box" or "density" a plot is returned. With method "t" or "wilcox", the text output from the test are returned.

**Examples**

```
measure_association_to_cell_properties(image_no_markers,
                                     celltypes = c("Tumour", "Immune1"),
                                     feature_colname = "Cell.Type",
                                     property = "Cell.Size",
                                     method = "box")
measure_association_to_cell_properties(image_no_markers,
                                     celltypes = c("Tumour", "Immune2"),
                                     feature_colname="Cell.Type",
                                     property = "Cell.Size",
                                     method = "t")
```

---

`mixing_score_summary` *Calculate the (normalised) mixing score for interested cell types*

---

**Description**

Produces a data.frame with mixing scores of input reference and target cells from a SpatialExperiment object. It calculates reference-target interactions and reference-reference interactions based on a radius. It derives the mixing score and the normalised mixing score. Function returns NA if the mixing score is being calculated between cells of the same type.

**Usage**

```
mixing_score_summary(
  spe_object,
  reference_celltype,
  target_celltype,
  radius = 20,
  feature_colname
)
```

**Arguments**

spe_object	SpatialExperiment object in the form of the output of <code>format_image_to_spe</code> .
reference_celltype	String Vector. Cell types of the reference cells.
target_celltype	String Vector. Cell types of the target cells.
radius	The maximum radius around a reference cell type for another cell to be considered an interaction.
feature_colname	String specifying the column with the desired cell type annotations.

**Details**

The mixing score was originally defined as the number of immune-tumour interactions divided by the number of immune-immune interactions within a defined radius (Keren et al., 2018). The normalised mixing score normalises the immune-tumour interactions and immune-immune interactions within radius by the total number of immune-tumour and immune-immune interactions in the image, respectively. We have generalized this score to allow calculation of any two cell phenotypes defined by the user.

**Value**

A data.frame of cell numbers, number of cell interactions, mixing scores, and normalised mixing scores. If there are no reference or target cells found in the image, or there are no reference cells found within the specified radius of any reference cells, the returned (normalised) mixing scores will be NA. If there are no target cells found within the radius of any reference cells, the returned (normalised) mixing scores will be 0.

**Examples**

```
mixing_score_summary(SPIAT::defined_image, reference_celltype = "Tumour", target_celltype="Immune1",
radius = 50, feature_colname = "Cell.Type")
```

---

number\_of\_cells\_within\_radius  
*Number of cells within a radius*

---

**Description**

Calculates the number of cells of a target cell type within a pre-defined radius around cells of a reference cell type.



**Usage**

```
number_of_cells_within_radius(
  spe_object,
  reference_celltype,
  target_celltype,
  radius = 20,
  feature_colname
)
```

**Arguments**

`spe_object` SpatialExperiment object in the form of the output of `format_image_to_spe`.

`reference_celltype` String. Cell type to be used for reference cells.

`target_celltype` String. Cell type to be used for target cells.

`radius` Numeric. Radius around the reference cells.

`feature_colname` String specifying the column with the desired cell type annotations.

**Value**

A list of dataframes with the number of target cells of each of the reference cells

**Examples**

```
n_in_radius <- number_of_cells_within_radius(SPIAT::defined_image,
  reference_celltype = "Tumour", target_celltype="Immune1", radius = 50,
  feature_colname = "Cell.Type")
```

---

```
plot_average_intensity
      plot_average_intensity
```

---

**Description**

Takes in a vector or radii and calculates the average intensity of a target marker using `average_intensity` function. It plots the intensity level as a line graph.

**Usage**

```
plot_average_intensity(spe_object, reference_marker, target_marker, radii)
```

**Arguments**

spe\_object      SpatialExperiment object in the form of the output of [format\\_image\\_to\\_spe](#).  
reference\_marker  
                  String specifying the reference marker.  
target\_marker    String specifying the marker to calculate its average intensity.  
radii            Numeric Vector specifying the search radius around reference cells.

**Value**

A plot is returned

**Examples**

```
plot_average_intensity(SPIAT::simulated_image, reference_marker="Immune_marker3",
target_marker="Immune_marker2", c(30, 35, 40, 45, 50, 75, 100))
```

---

plot\_cell\_categories    *plot\_cell\_categories*

---

**Description**

Produces a scatter plot of the cells of their x-y positions in the tissue. Cells are coloured categorically by phenotype. Cells not part of the phenotypes of interest will be coloured "lightgrey".

**Usage**

```
plot_cell_categories(
  spe_object,
  categories_of_interest = NULL,
  colour_vector = NULL,
  feature_colname = "Cell.Type",
  cex = 1,
  layered = FALSE
)
```

**Arguments**

spe\_object      SpatialExperiment object in the form of the output of [format\\_image\\_to\\_spe](#).  
categories\_of\_interest  
                  Vector of cell categories to be coloured.  
colour\_vector    Vector specifying the colours of each cell phenotype.  
feature\_colname  
                  String specifying the column the cell categories belong to.  
cex              Numeric. The size of the plot points. Default is 1.  
layered          Boolean. Whether to plot the cells layer by layer (cell categories). By default is FALSE.

**Value**

A plot is returned

**Examples**

```
categories_of_interest <- c("Tumour", "Immune1", "Immune2", "Immune3")
colour_vector <- c("red", "darkblue", "blue", "darkgreen")
plot_cell_categories(SPIAT::defined_image, categories_of_interest, colour_vector,
feature_colname = "Cell.Type")
```

---

```
plot_cell_distances_violin
      plot_cell_distances_violin
```

---

**Description**

Plots distances between cells as a violin plot

**Usage**

```
plot_cell_distances_violin(cell_to_cell_dist)
```

**Arguments**

cell\_to\_cell\_dist

Data.frame containing the distance output between cell types. The functions that generate the distances can be [calculate\\_minimum\\_distances\\_between\\_celltypes](#) and [calculate\\_pairwise\\_distances\\_between\\_celltypes](#).

**Value**

A plot is returned

**Examples**

```
distances <- calculate_pairwise_distances_between_celltypes(SPIAT::defined_image,
cell_types_of_interest = c("Immune1", "Immune2"), feature_colname="Cell.Type")
plot_cell_distances_violin(distances)
```

---

`plot_cell_marker_levels`  
*plot\_cell\_marker\_levels*

---

**Description**

Produces a scatter plot of the level of a marker in each cell. The level of the marker in all cells is shown, at x-y positions, no matter if cells are phenotyped as being positive or negative for the particular marker.

**Usage**

```
plot_cell_marker_levels(spe_object, marker)
```

**Arguments**

<code>spe_object</code>	SpatialExperiment object in the form of the output of <a href="#">format_image_to_spe</a> .
<code>marker</code>	String. Marker to plot.

**Value**

A plot is returned

**Examples**

```
plot_cell_marker_levels(SPIAT::simulated_image, "Immune_marker1")
```

---

`plot_cell_percentages` *plot\_cell\_percentages*

---

**Description**

Plots cells proportions as barplots.

**Usage**

```
plot_cell_percentages(  
  cell_proportions,  
  cells_to_exclude = NULL,  
  cellprop_colname = "Proportion_name"  
)
```

**Arguments**

cell\_proportions  
Data Frame. Output from [calculate\\_cell\\_proportions](#).

cells\_to\_exclude  
String Vector. Markers to exclude.

cellprop\_colname  
String. Column to use for y axis names. Default is "Proportion\_name".

**Value**

A plot is returned

**Examples**

```
p_cells <- calculate_cell_proportions(SPIAT::simulated_image)
plot_cell_percentages(p_cells)
```

---

plot\_composition\_heatmap  
*plot\_composition\_heatmap*

---

**Description**

Produces a heatmap showing the marker percentages within each cluster and the cluster sizes.

**Usage**

```
plot_composition_heatmap(
  composition,
  pheno_to_exclude = NULL,
  log_values = FALSE,
  feature_colname
)
```

**Arguments**

composition Data.frame. Output from [composition\\_of\\_neighborhoods](#).

pheno\_to\_exclude  
String Vector of phenotype to exclude.

log\_values Boolean. TRUE if the percentages should be logged (base 10).

feature\_colname  
String. Column with cell types.

**Value**

A plot is returned

## Examples

```
neighborhoods <- identify_neighborhoods(image_no_markers, method = "hierarchical",
min_neighborhood_size = 100, cell_types_of_interest = c("Immune", "Immune1", "Immune2"),
radius = 50, feature_colname = "Cell.Type")
neighborhoods_vis <- composition_of_neighborhoods(neighborhoods, feature_colname="Cell.Type")
plot_composition_heatmap(neighborhoods_vis, feature_colname="Cell.Type")
```

---

plot\_distance\_heatmap *plot\_distance\_heatmap*

---

## Description

Takes the output of `cell_distances` and plot the distances as a heatmap.

## Usage

```
plot_distance_heatmap(phenotype_distances_result, metric = "mean")
```

## Arguments

<code>phenotype_distances_result</code>	Dataframe output from <code>'calculate_summary_distances_between_celltypes'</code> or <code>'calculate_minimum_distances_between_celltypes'</code> .
<code>metric</code>	Metric to be plotted. One of "mean", "std.dev", "median", "min" or "max".

## Value

A plot is returned

## Examples

```
dists <- calculate_pairwise_distances_between_celltypes(SPIAT::defined_image,
cell_types_of_interest = c("Tumour", "Immune1"), feature_colname = "Cell.Type")
summary_distances <- calculate_summary_distances_between_celltypes(dists)
plot_distance_heatmap(summary_distances)
```

---

```
plot_marker_level_heatmap
      plot_marker_level_heatmap
```

---

### Description

Blurs the image by splitting the images into small squares. The marker levels are then averaged within each square. All cells are considered, regardless of phenotype status.

### Usage

```
plot_marker_level_heatmap(spe_object, num_splits, marker)
```

### Arguments

spe_object	SpatialExperiment object in the form of the output of <a href="#">format_image_to_spe</a> .
num_splits	Integer specifying the blurring level (number of splits) for the image. Higher numbers result in higher resolution.
marker	String. Marker to plot.

### Value

A plot is returned

### Examples

```
plot_marker_level_heatmap(SPIAT::simulated_image, num_splits = 100, "Tumour_marker")
```

---

```
predict_phenotypes      predict_phenotypes
```

---

### Description

Predicts cell phenotypes based on marker intensity levels. If no prior cell phenotypes are available, it adds the phenotypes to the SpatialExperiment object used as input. If reference cell phenotypes are available, it produces a density plot showing predicted cutoff of a positive reading for marker intensity and it returns a dataframe containing the predicted intensity status for a particular marker.

**Usage**

```

predict_phenotypes(
  spe_object,
  thresholds = NULL,
  tumour_marker,
  baseline_markers,
  nuclear_marker = NULL,
  reference_phenotypes = FALSE,
  markers_to_phenotype = NULL,
  plot_distribution = TRUE
)

```

**Arguments**

**spe\_object** SpatialExperiment object in the form of the output of [format\\_image\\_to\\_spe](#).

**thresholds** (Optional) Numeric Vector specifying the cutoff of a positive reading. The order must match the marker order, and it should be NA for DAPI.

**tumour\_marker** String containing the tumour\_marker used for the image. If tumor cells are known, annotate tumor cells as 1 and non-tumor cells as 0, and include the rowname.

**baseline\_markers** String Vector. Markers not found on tumour cells to refine the threshold used for tumour cell phenotyping.

**nuclear\_marker** String. Nuclear marker used.

**reference\_phenotypes** Boolean. TRUE or FALSE value whether there are reference phenotypes for the sample obtained by the user through other means (e.g. HALO or InForm). If there are reference phenotypes available, a matrix of predicted phenotypes, intensities, and reference phenotypes will be returned, which can be used as input to "marker\_prediction\_plot". If no reference phenotype available, the result of the function will be added to the spe object used in the input. Note that if a reference phenotype is to be used, the phenotypes must be an explicit combination of positive markers (e.g. AMACR,PDL1), as opposed to descriptive (PDL1+tumour cells).

**markers\_to\_phenotype** String Vector. Markers to be included in the phenotyping. If NULL, then all markers will be used. DAPI needs to be excluded.

**plot\_distribution** Boolean. If TRUE, plots of the marker intensities distributions and cutoffs are plotted.

**Value**

An updated spe object with cell phenotypes or a data.frame of predicted phenotypes



## Examples

```
# keep the original phenotypes
predicted_result <- predict_phenotypes(spe_object = simulated_image, thresholds = NULL,
tumour_marker = "Tumour_marker",baseline_markers = c("Immune_marker1", "Immune_marker2",
"Immune_marker3", "Immune_marker4"), reference_phenotypes = TRUE)
# update the predicted phenotypes
predicted_spe_image <- predict_phenotypes(spe_object = simulated_image, thresholds = NULL,
tumour_marker = "Tumour_marker",baseline_markers = c("Immune_marker1", "Immune_marker2",
"Immune_marker3", "Immune_marker4"), reference_phenotypes = FALSE)
```

---

R\_BC

*The ratio of border cell count to cluster cell count*

---

## Description

Calculates the ratio of the bordering cell count and the total to-be-clustered cell count in an image. The bordering cells are detected by the default [identify\\_bordering\\_cells](#) function. If the ratio is high, it means that most cells to be clustered are identified as bordering cells. This means there is no clear clusters.

## Usage

```
R_BC(spe_object, cell_type_of_interest, feature_colname)
```

## Arguments

`spe_object`        SpatialExperiment object in the form of the output of [format\\_image\\_to\\_spe](#).  
`cell_type_of_interest`  
                    String. The cell type that the user wants to determine a cluster of.  
`feature_colname`  
                    String. The column that contains the cell type to be clustered.

## Value

A number is returned.

## Examples

```
R_BC(SPIAT::defined_image, cell_type_of_interest = "Tumour", "Cell.Type")
```

---

select\_celltypes      *select\_celltypes*

---

### Description

Select cell types to keep or exclude in the analysis. The output of this function also includes the original image size and cell count.

### Usage

```
select_celltypes(  
  spe_object,  
  celltypes,  
  feature_colname = "Phenotype",  
  keep = TRUE  
)
```

### Arguments

spe_object	SpatialExperiment object in the form of the output of <a href="#">format_image_to_spe</a> .
celltypes	String Vector of celltypes of keep or exclude.
feature_colname	String. The column that has the interested cell types. If the cells ids are used to select cells, use "Cell.ID" for this arg.
keep	Boolean. TRUE if vector of 'celltypes' are the cells that are going to be kept, FALSE if they are to be removed.

### Value

A SpatialExperiment object is returned. The original image size and cell count can be accessed by 'attr(slim\_spe, "original\_cell\_number")' and 'attr(slim\_spe, "range\_of\_coords")', where 'slim\_spe' is the output of this function.

### Examples

```
data_subset <- select_celltypes(SPIAT::simulated_image,  
  celltypes = c("Tumour_marker", "Immune_marker1", "Immune_marker2",  
  "Immune_marker3", "Immune_marker4"),  
  feature_colname = "Phenotype", keep=TRUE)  
attr(data_subset, "original_cell_number") #cell number in the original image  
attr(data_subset, "range_of_coords")  
dim(data_subset)[2] # this is the new image cell number
```

---

simulated_image	<i>SPE object of a formatted image (simulated by 'spaSim' package)</i>
-----------------	--

---

**Description**

A dataset that contains a formatted spe object with cell ids and phenotypes in 'colData()' and marker intensities in 'assays()'.

**Usage**

```
simulated_image
```

**Format**

An SpatialExperiment object. Assay contains 5 rows (markers) and 4951 columns (cells); colData contains 4951 rows (cells) and 3 columns.

**See Also**

[defined\\_image](#) [image\\_no\\_markers](#)

---

SPIAT	<i>SPIAT: A package for spatial image analysis of tissues</i>
-------	---

---

**Description**

SPIAT has six modules including reading spatial data, basic analysis and visualization, calculation of cell colocalization, the automated characterization of clinically relevant immune spatial patterns, measuring spatial heterogeneity and neighborhood analysis. It uses 'SpatialExperiment' as the base class of analysis.

# Index

- \* **datasets**
  - defined\_image, 18
  - image\_no\_markers, 33
  - simulated\_image, 51
- AUC\_of\_cross\_function, 3
- average\_marker\_intensity\_within\_radius,  
4
- average\_minimum\_distance, 5
- average\_nearest\_neighbor\_index, 5
- average\_percentage\_of\_cells\_within\_radius,  
6
  
- calculate\_cell\_proportions, 7, 45
- calculate\_cross\_functions, 3, 8, 18
- calculate\_distance\_to\_margin, 9
- calculate\_entropy, 10
- calculate\_minimum\_distances\_between\_celltypes,  
11, 15, 43
- calculate\_pairwise\_distances\_between\_celltypes,  
11, 15, 43
- calculate\_percentage\_of\_grids, 12
- calculate\_proportions\_of\_cells\_in\_structure,  
13
- calculate\_spatial\_autocorrelation, 14
- calculate\_summary\_distances\_between\_celltypes,  
14
- calculate\_summary\_distances\_of\_cells\_to\_borders,  
15
- composition\_of\_neighborhoods, 16, 45
- compute\_gradient, 17
- crossing\_of\_crossK, 17
  
- define\_celltypes, 19
- define\_structure, 20
- defined\_image, 18, 34, 51
- dimensionality\_reduction\_plot, 21
  
- entropy\_gradient\_aggregated, 22
  
- format\_cellprofiler\_to\_spe, 23, 28
  
- format\_codex\_to\_spe, 24, 28
- format\_colData\_to\_spe, 25
- format\_halo\_to\_spe, 25, 28
- format\_image\_to\_spe, 4–8, 10–13, 17, 19,  
22, 27, 31–35, 37–42, 44, 47–50
- format\_inform\_to\_spe, 28, 29
- format\_spe\_to\_ppp, 30
  
- grid\_metrics, 12, 14, 31
  
- identify\_bordering\_cells, 31, 49
- identify\_neighborhoods, 16, 32
- image\_no\_markers, 18, 33, 51
- image\_splitter, 34
  
  
- marker\_intensity\_boxplot, 35
- marker\_prediction\_plot, 36
- marker\_surface\_plot, 36
- marker\_surface\_plot\_stack, 37
- measure\_association\_to\_cell\_properties,  
38
- mixing\_score\_summary, 39
- moran, 14
  
- number\_of\_cells\_within\_radius, 40
  
- plot\_average\_intensity, 41
- plot\_cell\_categories, 42
- plot\_cell\_distances\_violin, 43
- plot\_cell\_marker\_levels, 44
- plot\_cell\_percentages, 44
- plot\_composition\_heatmap, 45
- plot\_distance\_heatmap, 46
- plot\_marker\_level\_heatmap, 47
- predict\_phenotypes, 36, 47
  
- R\_BC, 49
  
- select\_celltypes, 50
- simulated\_image, 18, 34, 51
- SPIAT, 51