Package: SpatialArtifacts (via r-universe)

November 25, 2026

Title Identification and Classification of Spatial Artifacts in Visium and Visium HD Data

Version 1.2.0

Description SpatialArtifacts provides a data-driven two-step workflow to identify, classify, and handle spatial artifacts in spatial transcriptomics data. The package combines median absolute deviation (MAD)-based outlier detection with morphological image processing (fill, outline, and star patterns) to detect edge and interior artifacts. It supports multiple platforms including 10x Genomics Visium (standard and HD), allowing for consistent quality control across different spatial resolutions.

Depends R (>= 4.4.0)

License Artistic-2.0

URL https://github.com/CambridgeCat13/SpatialArtifacts

BugReports https://github.com/CambridgeCat13/SpatialArtifacts/issues

biocViews Software, Spatial, Transcriptomics, QualityControl, DataImport, WorkflowStep, Classification

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

SystemRequirements quarto, GDAL (>= 2.0.1), GEOS (>= 3.4.0), PROJ (>= 4.8.0)

VignetteBuilder knitr

Imports SpatialExperiment, SummarizedExperiment, S4Vectors, scuttle, dplyr, terra, stats, methods

Suggests BiocStyle, knitr, rmarkdown, BiocCheck, ggplot2, patchwork, testthat (>= 3.0.0)

Config/testthat/edition 3

LazyData true

Config/pak/sysreqs libgdal-dev gdal-bin libgeos-dev libmagick++-dev gsfonts libicu-dev libssl-dev libproj-dev libsqlite3-dev zlib1g-dev

Repository Bioconductor 3.24

Date/Publication 2026-11-23 22:55:35 UTC

RemoteUrl https://github.com/bioc/SpatialArtifacts

RemoteRef RELEASE_3_24

RemoteSha 17efb219f72f83885fe2b8cb19c3bef3bc7091c4

Contents

SpatialArtifacts-package 2
classifyEdgeArtifacts 3
clumpEdges 6
detectEdgeArtifacts 7
detectEdgeArtifacts_Visium 8
detectEdgeArtifacts_VisiumHD 10
focal_transformations 13
focal_transformations_terra 14
problemAreas 15
problemAreas_WithMorphology_terra 17
spe_vignette 18

Index **20**

`SpatialArtifacts-package`

SpatialArtifacts: Detect and Classify Spatial Artifacts

Description

`SpatialArtifacts` provides a robust, data-driven two-step workflow to identify, classify, and handle spatial artifacts in spatial transcriptomics data across multiple platforms including 10x Genomics Visium (Standard and HD). It combines median absolute deviation (MAD)-based outlier detection with morphological image processing to flag problematic regions such as edge dryspots and interior artifacts caused by incomplete reagent coverage.

Main Functions

- `detectEdgeArtifacts`: The primary wrapper function to detect potential artifact spots. Automatically routes to platform-specific methods based on the `platform` argument. Outputs three columns to `colData`: `*_edge`, `*_problem_id`, and `*_problem_size`.
- `classifyEdgeArtifacts`: Hierarchically classifies detected artifacts by location (edge vs. interior) and size (large vs. small). Outputs a single `*_classification` column.

Typical Workflow

```
# Step 1: Detect artifacts
spe <- detectEdgeArtifacts(spe, platform = "visium", qc_metric = "sum_gene")

# Step 2: Classify artifacts
spe <- classifyEdgeArtifacts(spe, min_spots = 20)
```

Platform-Specific Usage

`detectEdgeArtifacts` requires users to specify their platform:

- **Standard Visium** (`platform = "visium"`): Uses hexagonal grid layout. The default `shifted = FALSE` is correct for standard Space Ranger `array_row`/`array_col` outputs.
- **Visium HD** (`platform = "visiumhd"`): Uses square grid layout. Requires the `resolution` parameter (`"16um"` or `"8um"`). Parameters are specified in physical units (micrometers).

Input Data

All functions accept a `SpatialExperiment` object. QC metrics (e.g., library size, detected genes) should be precomputed, for example using `addPerCellQCMetrics`.

Author(s)

Maintainer: Harriet Jiali He <jhe46@jh.edu> (ORCID)

Authors:

- Jacqueline R. Thompson <jthom338@jh.edu>
- Michael Totty <mtotty2@jh.edu>
- Stephanie C. Hicks <shicks19@jhu.edu> (ORCID) [funder]

See Also

Useful links:

- https://github.com/CambridgeCat13/SpatialArtifacts
- Report bugs at https://github.com/CambridgeCat13/SpatialArtifacts/issues

---

`classifyEdgeArtifacts` *Classify Edge Artifacts into Hierarchical Categories*

Description

Classifies detected artifacts based on their location (edge vs. interior) and size (large vs. small). This function works downstream of `detectEdgeArtifacts()`.

Usage

```
classifyEdgeArtifacts(
  spe,
  qc_metric = "sum_umi",
  samples = "sample_id",
  min_spots = 20,
  name = "edge_artifact",
  exclude_slides = NULL
)
```

Arguments

spe	A SpatialExperiment object that has been processed with `detectEdgeArtifacts()`.
qc_metric	Character string specifying the QC metric column used for validation (default: "sum_umi"). Note: This column must exist, but is not directly used for classification logic.
samples	Character string specifying the sample ID column name (default: "sample_id").
min_spots	Minimum number of spots for an artifact to be classified as "large" (default: 20).
name	Character string matching the `name` argument used in `detectEdgeArtifacts()` (default: "edge_artifact").
exclude_slides	Character vector of slide IDs to exclude from edge detection (default: NULL). Spots on these slides will be forced to FALSE for edge artifacts.

Details

Parameter Recommendations:

The `min_spots` threshold should scale with platform resolution to represent similar physical artifact sizes:

- **Standard Visium (55µm bins):** `min_spots` = `20-40`
  - Physical area: ~0.06-0.12 mm²
  - Rationale: Artifacts <20 spots likely represent isolated noise
  - Typical edge artifacts: 50-200 spots
- **VisiumHD 16µm bins:** `min_spots` = `100-200`
  - Physical area: ~0.026-0.051 mm² (comparable to Visium)
  - Rationale: Higher density requires proportionally higher threshold
  - Typical edge artifacts: 500-2000 bins
- **VisiumHD 8µm bins:** `min_spots` = `400-800`
  - Physical area: ~0.026-0.051 mm² (comparable to Visium)
  - Rationale: 4× density of 16µm bins requires 4× threshold
  - Typical edge artifacts: 2000-8000 bins

Practical Guideline: For VisiumHD data, start with: `min_spots` = `20 × (55µm / bin_size)²`

This formula maintains constant physical area thresholds across resolutions.

Context-Specific Tuning:

- Increase `min_spots` for noisy data (prevents over-flagging small clusters)
- Decrease `min_spots` for high-quality data (captures smaller genuine artifacts)
- Visualize intermediate results to validate threshold appropriateness

Value

A SpatialExperiment object with additional classification columns in `colData`:

- `[name]_true_edges`: Logical, indicating edge artifacts after applying slide exclusions.
- `[name]_classification`: Character, containing hierarchical categories:
  - `"not_artifact"`: Normal, high-quality spots.
  - `"large_edge_artifact"`: Clusters > min_spots touching the slide boundary.
  - `"small_edge_artifact"`: Clusters <= min_spots touching the slide boundary.
  - `"large_interior_artifact"`: Clusters > min_spots in the tissue interior.
  - `"small_interior_artifact"`: Clusters <= min_spots in the tissue interior.

See Also

`detectEdgeArtifacts`

Examples

```
library(SpatialExperiment)
library(S4Vectors)

# --- Create a Mock SPE with "Detected" Results ---
# We simulate the output that detectEdgeArtifacts() would produce
n_spots <- 5
spe <- SpatialExperiment(
  colData = DataFrame(
    sample_id = "sample01",
    sum_umi = rep(100, n_spots),
    edge_artifact_edge = c(TRUE, TRUE, FALSE, FALSE, FALSE),
    edge_artifact_problem_id = c("Cluster1", "Cluster1", "Cluster2", "Cluster3", NA),
    edge_artifact_problem_size = c(50, 50, 10, 30, 0)
  )
)

# Review the mock scenarios:
# Spot 1: Edge=TRUE, Size=50 -> Expect "large_edge_artifact"
# Spot 2: Edge=TRUE, Size=50 -> Expect "large_edge_artifact"
# Spot 3: Edge=FALSE, Size=10 -> Expect "small_interior_artifact"
# Spot 4: Edge=FALSE, Size=30 -> Expect "large_interior_artifact"
# Spot 5: Edge=FALSE, Size=0  -> Expect "not_artifact"

# --- Run Classification ---
# Set threshold to 20 to separate large/small
spe <- classifyEdgeArtifacts(spe, min_spots = 20, name = "edge_artifact")
table(spe$edge_artifact_classification)
colData(spe)[, c("edge_artifact_problem_size", "edge_artifact_classification")]
```

`clumpEdges` *Detect edge dryspots using spatial clustering*

Description

Identifies clusters of low-quality spots that touch tissue boundaries, indicating potential edge dryspot artifacts from incomplete reagent coverage.

Usage

```
clumpEdges(
  .xyz,
  offTissue,
  shifted = FALSE,
  edge_threshold = 0.75,
  min_cluster_size = 40
)
```

Arguments

`.xyz`	Data frame with spot coordinates and QC metrics. Must contain: • Column 1: array_row coordinates • Column 2: array_col coordinates • Column 3: Binary outlier indicator (TRUE/FALSE or 1/0) • Row names: Spot identifiers
`offTissue`	Character vector of spot identifiers that are off-tissue
`shifted`	Logical indicating whether to apply coordinate adjustment for hexagonal arrays (default: FALSE)
`edge_threshold`	Numeric value between 0 and 1 specifying the minimum proportion of border coverage required for edge detection (default: 0.75)
`min_cluster_size`	Numeric value for minimum cluster size in morphological operations (default: 40)

Details

The function performs the following steps:

1. Converts spot coordinates to raster format
2. Applies morphological transformations to connect outlier regions
3. Identifies connected components (clusters)
4. Checks each tissue border (N, S, E, W) for cluster coverage
5. Returns spots from clusters that exceed edge_threshold on any border

Value

Character vector of spot identifiers classified as edge dryspots

Examples

```
# 1. Create a 5x5 grid of mock spot data
spot_data <- data.frame(
  array_row = rep(1:5, each = 5),
  array_col = rep(1:5, times = 5),
  outlier = FALSE
)
rownames(spot_data) <- paste0("spot", 1:25)

# 2. Create an "edge artifact" by flagging the first row as outliers
spot_data$outlier[spot_data$array_row == 1] <- TRUE

# 3. Define off-tissue spots (none in this simple case)
offTissue <- character(0)

# 4. Detect edge dryspots
edge_spots <- clumpEdges(
  spot_data,
  offTissue = offTissue,
  edge_threshold = 0.5,
  min_cluster_size = 3
)
print(edge_spots)
```

---

detectEdgeArtifacts *Unified Edge Artifact Detection*

Description

A convenient wrapper that routes to platform-specific edge detection methods.

Usage

```
detectEdgeArtifacts(spe, platform = c("visium", "visiumhd"), ...)
```

Arguments

spe	A SpatialExperiment object.
platform	Character string: "visium" or "visiumhd" (case insensitive).
...	Additional arguments passed to the specific function: • For **Visium**: edge_threshold, min_cluster_size, etc. • For **VisiumHD**: resolution (REQUIRED), buffer_width_um, etc.

Value

A SpatialExperiment object with artifact detection columns (_edge, _problem_id, _problem_size) added to colData.

Examples

```
# Load example data
data(spe_vignette)

# 1. Standard Visium example (runnable)
spe_visium <- detectEdgeArtifacts(spe_vignette,
platform = "visium", qc_metric = "sum", batch_var = "sample_id")
# 2. Visium HD example (wrapped in  to avoid execution without HD data)

# spe_hd <- detectEdgeArtifacts(spe_hd_example, platform = "visiumhd", resolution = "16um")
```

---

detectEdgeArtifacts_Visium

Detect edge artifacts in spatial transcriptomics data

Description

This function identifies edge artifacts and problem areas in spatial transcriptomics data by analyzing QC metrics and spatial patterns.

Usage

```
detectEdgeArtifacts_Visium(
  spe,
  qc_metric = "sum_gene",
  samples = "sample_id",
  mad_threshold = 3,
  edge_threshold = 0.75,
  min_cluster_size = 40,
  shifted = FALSE,
  batch_var = "both",
  name = "edge_artifact",
  verbose = TRUE,
  keep_intermediate = FALSE
)
```

Arguments

spe	A SpatialExperiment object containing spatial transcriptomics data
qc_metric	Character string specifying the QC metric column name to analyze (default: "sum_gene")

samples	Character string specifying the sample ID column name (default: "sample_id")
mad_threshold	Numeric value for MAD threshold for outlier detection (default: 3)
edge_threshold	Numeric threshold for edge detection (default: 0.75)
min_cluster_size	Minimum cluster size for morphological cleaning (default: 40)
shifted	Logical indicating whether to apply coordinate adjustment for hexagonal arrays (default: FALSE)
batch_var	Character specifying batch variable for outlier detection ("slide", "sample_id", or "both", default: "both")
name	Character string for naming output columns (default: "edge_artifact")
verbose	Logical indicating whether to print progress messages (default: TRUE)
keep_intermediate	Logical indicating whether to keep intermediate outlier detection columns (default: FALSE)

Value

A SpatialExperiment object with additional columns in colData:

\link[terra:names]{terra::name}_edge	Logical indicating spots identified as edges
\link[terra:names]{terra::name}_problem_id	Character identifying problem area clusters
\link[terra:names]{terra::name}_problem_size	Numeric size of problem area clusters

Examples

```
library(SummarizedExperiment)
library(SpatialExperiment)
library(S4Vectors)

# Create a minimal mock SpatialExperiment (4x4 grid with edge artifact)
set.seed(123)
n_spots <- 16
coords <- expand.grid(row = 1:4, col = 1:4)

# Simulate counts with lower values at edges (top row)
mock_counts <- rpois(n_spots, lambda = 500)
mock_counts[1:4] <- rpois(4, lambda = 50) # Edge artifact

spe_mock <- SpatialExperiment::SpatialExperiment(
  assays = list(counts = matrix(rpois(n_spots * 10, lambda = 5),
    nrow = 10, ncol = n_spots
  )),
  colData = DataFrame(
    in_tissue = rep(TRUE, n_spots),
    sum_gene = mock_counts,
```

```
    sum_umi = mock_counts, # Add sum_umi for classify function
    sample_id = "mock_sample",
    slide = "mock_slide",
    array_row = coords$row,
    array_col = coords$col
  ),
  spatialCoords = as.matrix(coords)
)

colnames(spe_mock) <- paste0("spot_", seq_len(n_spots))
rownames(spe_mock) <- paste0("gene_", 1:10)

# Detect edge artifacts
spe_detected <- detectEdgeArtifacts_Visium(
  spe_mock,
  qc_metric = "sum_gene",
  samples = "sample_id",
  mad_threshold = 3,
  min_cluster_size = 1,
  name = "edge_artifact"
)

# Check detection results
table(spe_detected$edge_artifact_edge)
head(colData(spe_detected)[, c(
  "edge_artifact_edge",
  "edge_artifact_problem_id"
)])
```

`detectEdgeArtifacts_VisiumHD` *Detect Edge Artifacts in VisiumHD Data*

Description

Detect Edge Artifacts in VisiumHD Data

Usage

```
detectEdgeArtifacts_VisiumHD(
  spe,
  resolution,
  qc_metric = "sum_gene",
  samples = "sample_id",
  mad_threshold = 3,
  buffer_width_um = 80,
  min_cluster_area_um2 = 1280,
  batch_var = "sample_id",
  col_x = "array_col",
```

```
  col_y = "array_row",
  name = "edge_artifact",
  verbose = TRUE,
  keep_intermediate = FALSE
)
```

Arguments

spe	SpatialExperiment object
resolution	Resolution: "8um" or "16um" (REQUIRED)
qc_metric	QC metric column (default: "sum_gene")
samples	Sample ID column (default: "sample_id")
mad_threshold	MAD threshold (default: 3)
buffer_width_um	Buffer zone width in micrometers (default: 80) Approximately 10 bins at 8um resolution, 5 bins at 16um resolution
min_cluster_area_um2	Minimum cluster area in um^2 (default: 1280) Approximately 20 bins at 8um resolution, 5 bins at 16um resolution Default based on 16um standard (5 bins = reasonable minimum cluster)
batch_var	Batch variable (default: "sample_id")
col_x	X coordinate column (default: "array_col")
col_y	Y coordinate column (default: "array_row")
name	Output column prefix (default: "edge_artifact")
verbose	Print progress (default: TRUE)
keep_intermediate	Keep intermediate columns (default: FALSE)

Details

IMPORTANT: This function uses array_col/array_row (bin indices), NOT pixel coordinates. This is much more memory efficient.

Buffer width and cluster size are specified in physical units (um, um^2) and automatically converted to bins based on resolution:

- 8um resolution: 1 bin = 8*8 um = 64 um^2
- 16um resolution: 1 bin = 16*16 um = 256 um^2

Default parameters are designed for 16um resolution:

- buffer_width_um = 80 um -> 5 bins at 16um, 10 bins at 8um
- min_cluster_area_um2 = 1280 um^2 -> 5 bins at 16um, 20 bins at 8um

Value

A SpatialExperiment object with artifact detection columns added to colData:

- [name]_edge: Logical, indicates if a bin is an outlier within the buffer zone.
- [name]_problem_id: Character, unique ID for each detected interior problem area.
- [name]_problem_size: Numeric, the number of bins within each problem area cluster.

Examples

```
library(SpatialExperiment)
library(S4Vectors)

# 1. Runnable Example: Mock Data (Try this!)

# Create a mock Visium HD dataset (20x20 grid, representing 320x320 um)
n_rows <- 20
n_cols <- 20
n_bins <- n_rows * n_cols
coords <- expand.grid(array_row = seq_len(n_rows), array_col = seq_len(n_cols))

# Simulate gene counts: Edge artifact (left 2 cols) has low counts
counts <- rep(100, n_bins)
is_edge <- coords$array_col <= 2
counts[is_edge] <- 10

# Create SpatialExperiment object
spe_hd <- SpatialExperiment(
  assays = list(counts = matrix(counts, nrow = 1, ncol = n_bins)),
  colData = DataFrame(
    sample_id = "mock_hd",
    in_tissue = rep(TRUE, n_bins),
    sum_gene = counts,
    array_row = coords$array_row,
    array_col = coords$array_col
  )
)

# Run detection for 16um resolution
# (Physical buffer 80um / 16um bin size = 5 bins. Artifact is 2 bins wide.)

colnames(spe_hd) <- paste0("spot_", seq_len(ncol(spe_hd)))
rownames(spe_hd) <- paste0("gene_", seq_len(nrow(spe_hd)))

spe_hd <- detectEdgeArtifacts_VisiumHD(
  spe_hd,
  resolution = "16um",
  qc_metric = "sum_gene"
)

# Check results
table(spe_hd$edge_artifact_edge)
```

```
# 2. Illustrative Examples (Concept only)
## Not run:
# Assuming 'spe' is a real SpatialExperiment object

# 8um data with defaults
# buffer_width_um = 80 -> 10 bins (80 / 8)
# min_cluster_area_um2 = 1280 -> 20 bins
spe <- detectEdgeArtifacts_VisiumHD(spe, resolution = "8um")

# 16um data with defaults
# buffer_width_um = 80 -> 5 bins (80 / 16)
# min_cluster_area_um2 = 1280 -> 5 bins
spe <- detectEdgeArtifacts_VisiumHD(spe, resolution = "16um")

# Custom parameters (physical units)
spe <- detectEdgeArtifacts_VisiumHD(
  spe,
  resolution = "16um",
  buffer_width_um = 100, # 100 $u$m buffer
  min_cluster_area_um2 = 2000 # 2000 $u$m^2 minimum
)

## End(Not run)
```

`focal_transformations` *Apply morphological transformations to identify connected outlier regions*

Description

Performs a series of focal operations to clean and connect outlier regions in spatial transcriptomics data through morphological operations.

Usage

```
focal_transformations(raster_object, min_cluster_size = 40)
```

Arguments

`raster_object`	A terra SpatRaster object with binary values where 1 indicates outlier spots and 0/NA indicates normal spots
`min_cluster_size`	Numeric value specifying the minimum size for isolated clusters. Clusters smaller than this threshold will be filled (default: 40)

Details

The function applies four sequential morphological operations:

1. 3x3 fill: Fills spots completely surrounded by outliers
2. 5x5 outline: Fills spots outlined by outliers in larger window
3. Star pattern: Fills spots with outliers in cardinal directions
4. Small cluster removal: Removes isolated normal regions below threshold

Value

A processed SpatRaster object with cleaned and connected outlier regions

Examples

```
library(terra)
# Create a 5x5 mock raster object with an outlier (1)
m <- matrix(0, nrow = 5, ncol = 5)
m[2, 2] <- 1 # A single outlier
r <- terra::rast(m) # Use terra instead of raster

# Apply morphological cleaning
r_cleaned <- focal_transformations(r, min_cluster_size = 3)

# See the original and cleaned values
print(terra::values(r))
print(terra::values(r_cleaned))
```

---

focal_transformations_terra

Morphological transformations using terra (optimized for VisiumHD)

Description

Terra-only implementation of focal operations for connecting outlier regions.

Usage

```
focal_transformations_terra(r, min_cluster_size = 5)
```

Arguments

r	A SpatRaster object with binary values (1=outlier, 0=normal).
min_cluster_size	Minimum size (in bins) for small hole removal.

Value

A SpatRaster object after applying focal transformations (fill, outline, star) and small hole removal.

Examples

```
# Create a dummy binary SpatRaster for demonstration
if (requireNamespace("terra", quietly = TRUE)) {
  r <- terra::rast(matrix(c(0, 0, 0, 0, 1, 0, 0, 0, 0), nrow = 3),
    extent = terra::ext(0, 3, 0, 3)
  )
  # Run the focal transformations
  # result <- focal_transformations_terra(r, min_cluster_size = 5)
}
```

---

problemAreas *Identify all problem areas in spatial transcriptomics data*

Description

Detects and characterizes all clusters of low-quality spots (problem areas) in tissue sections, including both edge and interior artifacts.

Usage

```
problemAreas(
  .xyz,
  offTissue,
  uniqueIdentifier = NA,
  shifted = FALSE,
  min_cluster_size = 40
)
```

Arguments

.xyz	Data frame with spot coordinates and QC metrics. Must contain: • Column 1: array_row coordinates • Column 2: array_col coordinates • Column 3: Binary outlier indicator (TRUE/FALSE or 1/0) • Row names: Spot identifiers
offTissue	Character vector of spot identifiers that are off-tissue
uniqueIdentifier	Character string used as prefix for cluster IDs. If NA, "X" will be used (default: NA)

`shifted` Logical indicating whether to apply coordinate adjustment for hexagonal arrays (default: FALSE)

`min_cluster_size` Numeric value for minimum cluster size in morphological operations (default: 40)

Details

This function identifies ALL connected components of outlier spots, not just those touching edges. Each cluster is assigned a unique ID and its size is calculated. This enables downstream filtering based on cluster characteristics.

Value

Data frame with the following columns:

- spotcode: Spot identifier
- clumpID: Unique cluster identifier (format: "prefix_number")
- clumpSize: Number of spots in the cluster

Returns empty data frame if no problem areas are found.

Examples

```
# 1. Create a 5x5 grid of mock spot data
spot_data <- data.frame(
  array_row = rep(1:5, each = 5),
  array_col = rep(1:5, times = 5),
  outlier = FALSE
)
rownames(spot_data) <- paste0("spot", 1:25)

# 2. Create an "artifact" by flagging a 2x2 area as outliers
spot_data$outlier[spot_data$array_row %in% 2:3 & spot_data$array_col %in% 2:3] <- TRUE

# 3. Define off-tissue spots
offTissue <- character(0)

# 4. Identify all problem areas
problem_df <- problemAreas(
  spot_data,
  offTissue = offTissue,
  uniqueIdentifier = "Sample1",
  min_cluster_size = 3
)
print(problem_df)
```

problemAreas_WithMorphology_terra *Detect problem areas in VisiumHD data using morphological processing*

Description

Terra-optimized implementation for VisiumHD. Uses morphological operations to connect outlier regions and identify connected components.

Usage

```
problemAreas_WithMorphology_terra(
  .xyz,
  uniqueIdentifier = NA,
  min_cluster_size = 5,
  resolution = "8um"
)
```

Arguments

.xyz	Data frame with columns: x, y, outlier (binary 0/1)
uniqueIdentifier	Character string for cluster ID prefix
min_cluster_size	Minimum cluster size in bins (default: 5)
resolution	VisiumHD resolution: "8um" or "16um" (default: "8um")

Value

Data frame with columns: spotcode, clumpID, clumpSize

Examples

```
library(terra)

# 1. Create a mock VisiumHD-like coordinate dataframe
# 10x10 grid
coords <- expand.grid(x = 1:10, y = 1:10)
.xyz <- data.frame(
  x = coords$x,
  y = coords$y,
  outlier = 0
)
rownames(.xyz) <- paste0("bin_", 1:100)

# 2. Create a "problem area" (cluster of outliers)
# Make a 3x3 block of outliers in the center
```

```
center_mask <- .xyz$x >= 4 & .xyz$x <= 6 & .xyz$y >= 4 & .xyz$y <= 6
.xyz$outlier[center_mask] <- 1

# 3. Run detection
clusters <- problemAreas_WithMorphology_terra(
  .xyz,
  uniqueIdentifier = "TEST",
  min_cluster_size = 2,
  resolution = "16um"
)

# Check results
print(clusters)
```

`spe_vignette` *Example SpatialExperiment for Vignettes*

Description

A lightweight `SpatialExperiment` object derived from a human hippocampus Visium sample, used for demonstrating the `SpatialArtifacts` artifact detection workflow.

Format

A `SpatialExperiment` object with 12,971 genes and 4,965 spots, containing one sample (V11L05-335_C1). The `colData` includes precomputed QC metrics (e.g., `sum`, `detected`) from `addPerCellQC`.

Details

The object was derived from human hippocampus Visium data (sample V11L05-335_C1) from the spatialHPC project (LIBD4035). To meet Bioconductor package size requirements (<5 MB), the object was subset to highly variable genes using `scran::getTopHVGs()`, image data was removed, and the counts matrix was stored as a sparse matrix with XZ compression. QC metrics in `colData` were computed prior to gene subsetting and remain accurate for the full spot set.

Value

A `SpatialExperiment` object.

Source

Derived from human hippocampus Visium data (sample V11L05-335_C1) from Thompson et al. (2025). The raw Space Ranger output was accessed internally via the spatialHPC project (LIBD4035) on the JHPCE cluster. The same dataset is publicly available via the `humanHippocampus2024` Bioconductor ExperimentHub package (https://bioconductor.org/packages/humanHippocampus2024). A script to reproduce this object is provided in `inst/scripts/make-data.R`.

References

Thompson, J.R., Nelson, E.D., Tippani, M. et al. (2025). An integrated single-nucleus and spatial transcriptomics atlas reveals the molecular landscape of the human hippocampus. *Nature Neuroscience* 28, 1990–2004. doi:10.1038/s41593025020220

Examples

```
data(spe_vignette)
spe_vignette
```

Index

addPerCellQC, *18*
addPerCellQCMetrics, *3*

classifyEdgeArtifacts, *2*, 3
clumpEdges, 6

detectEdgeArtifacts, *2, 3*, *5*, 7
detectEdgeArtifacts_Visium, 8
detectEdgeArtifacts_VisiumHD, 10

focal_transformations, 13
focal_transformations_terra, 14

problemAreas, 15
problemAreas_WithMorphology_terra, 17

SpatialArtifacts
        *(*SpatialArtifacts-package*)*, 2
SpatialArtifacts-package, 2
SpatialExperiment, *3*, *8*, *12*, *18*
SpatRaster, *14, 15*
spe_vignette, 18