Package 'stPipe'

November 12, 2026

Type Package

Title Upstream pre-processing for Sequencing-Based Spatial Transcriptomics

Version 1.2.0

biocViews ImmunoOncology, Software, Sequencing, RNASeq, GeneExpression, SingleCell, Visualization, SequenceMatching, Preprocessing, QualityControl, GenomeAnnotation, DataImport, Spatial, Transcriptomics, Clustering

Description This package serves as an upstream pipeline for pre-processing sequencing-based spatial transcriptomics data. Functions includes FASTQ trimming, BAM file reformatting, index building, spatial barcode detection, demultiplexing, gene count matrix generation with UMI deduplication, QC, and revelant visualization. Config is an essential input for most of the functions which aims to improve reproducibility.

Depends R (>= 4.5.0)

Imports basilisk, data.table, DropletUtils, dplyr, ggplot2, methods, pbmcapply, reticulate, rmarkdown, Rcpp, Rhtslib, Rsubread, Rtsne, Seurat, SeuratObject, scPipe, shiny, SummarizedExperiment, SingleCellExperiment, SpatialExperiment, stats, umap, yaml

LinkingTo Rcpp, Rhdf5lib, testthat, Rhtslib

SystemRequirements GNU make

License GPL-3

Encoding UTF-8

LazyData false

RoxygenNote 7.3.2

URL https://github.com/mritchielab/stPipe

BugReports https://github.com/mritchielab/stPipe/issues/new

Suggests knitr, plotly, BiocStyle, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

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

git_branch RELEASE_3_23

git_last_commit 88cc1dd

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-11-11

Author Yang Xu [aut, cre] (ORCID: <https://orcid.org/0009-0008-3274-6516>),
Callum Sargeant [aut],
Shian Su [aut],
Luyi Tian [aut],
Yunshun Chen [ctb],
Matthew Ritchie [ctb, fnd]

Maintainer Yang Xu <xu.ya@wehi.edu.au>

Contents

Run_Clustering 2
Run_Create_Obj 3
Run_HTML 4
Run_Interactive 6
Run_Loc_Match 7
Run_QC 8
Run_ST 10
Run_Visualization 11
stPipe 12

Index **13**

`Run_Clustering` *Perform Basic Clustering Algorithms*

Description

This function performs basic clustering using t-SNE and UMAP, either before or after the QC step and outputs visualizations results.

Usage

```
Run_Clustering(gene.count, matched.data, num_clusters = 5)
```

Arguments

`gene.count`	A data frame containing gene count data with gene IDs as row names.
`matched.data`	A data frame containing matched spatial coordinates with raw UMI counts.
`num_clusters`	Number of clusters during clustering. Default set to five.

Details

T-SNE (t-Distributed Stochastic Neighbor Embedding)

T-SNE is a popular dimensionality reduction technique used to project high-dimensional data into two or three dimensions. It optimizes for preserving the local structure of the data, meaning that points that are close to each other in high-dimensional space will remain close in the lower-dimensional representation. This method is widely used in single-cell RNA-seq and spatial transcriptomics to explore the heterogeneity of the data and identify cells or spatial regions with similar expression profiles.

UMAP (Uniform Manifold Approximation and Projection)

UMAP is another popular dimensionality reduction technique that aims to preserve both the local and global structures of the data, with a stronger emphasis on efficiency. It excels at handling large datasets, generating low-dimensional projections in less time compared to T-SNE. Unlike T-SNE, UMAP seeks to retain both local and global data structures, making it more suitable for capturing broad clustering patterns across the datasets.

Value

A list which contains interactive clustering visualization result with related data frames.

Examples

```
set.seed(123)
gene.count <- matrix(sample(0:100, 200 * 100, replace = TRUE), nrow = 200)
rownames(gene.count) <- paste0("Gene", seq_len(200))
colnames(gene.count) <- paste0("Spot", seq_len(100))
matched.data <- data.frame(
  spatial_name = paste0("Spot", seq_len(100)),
  x_coord = runif(100, 0, 10),
  y_coord = runif(100, 0, 10)
)
result <- Run_Clustering(
  gene.count = gene.count,
  matched.data = matched.data,
  num_clusters = 3
)
```

Run_Create_Obj *This function creates specified spatial transcriptomics data object for further personalized downstream analysis*

Description

This function creates specified spatial transcriptomics data object for further personalized downstream analysis

Usage

```
Run_Create_Obj(gene.matrix, matched.data, obj.type, tech, ss.radius = 3000)
```

Arguments

gene.matrix	Gene count matrix. This is usually obtained from 'Run_ST' function.
matched.data	Data frame which contains spatial localization matched with gene count matrix. This is usually obtained from 'Run_loc_match' function.
obj.type	Type of spatial transcriptomics object being created, can be 'Seurat', 'SpatialExperiment', or 'AnnData'.
tech	Type of spatial transcriptomics sequencing technology, can be "Visium", "Slideseq", "Curio-seeker", or "Stereoseq".
ss.radius	Optional. Radius for filtering SlideSeq or Curio-seeker spots for Seurat object. Default is 3000.

Details

Current mainstream analytic tools for downstream includes: Seurat, SpatialExperiment supported tools, and Squidpy. This function can help creates corresponding objects for further downstream analysis.

Value

Created spatial transcriptomics data object as required by 'obj.type'. set.seed(123) gene.count <- matrix(sample(0:100, 200*100, TRUE), nrow=200) rownames(gene.count) <- paste0('Gene', 1:200) colnames(gene.count) <- paste0('Spot', 1:100) matched.data <- data.frame(spatial_name = colnames(gene.count), barcode_sequence = paste0('BC', 1:100), X_coordinate = runif(100,0,10), Y_coordinate = runif(100,0,10), UMI_count = sample(50:200,100,TRUE)) seu <- Run_Create_Obj(gene.matrix = gene.count, matched.data = matched.data, obj.type = 'Seurat', tech = 'Slideseq')

Run_HTML *HTML report generation in RMarkDown format*

Description

This function generates a HTML report of stPipe upstream processing steps in Rmarkdown format. It extracts plots in the result directory and outputs them in the whole report with corresponding explanation.

Usage

```
Run_HTML(path)
```

Arguments

path	Path to where results are stored, including the ones obtained from 'Run_ST', 'Run_loc_match', 'Run_Vis', and 'Run_Clustering'.

Value

Generated HTML report in RMarkDown format.

Examples

```
## @example
temp_dir <- tempdir()
dummy_gene_count <- data.frame(GeneA = c(10, 20), GeneB = c(5, 15))
rownames(dummy_gene_count) <- c("spot1", "spot2")
write.csv(dummy_gene_count, file = file.path(temp_dir, "gene_count.csv"), row.names = TRUE)
png_files <- c("Mapping_statistics_plot.png", "UMI_duplication_plot.png",
             "Barcode_demultiplexing_plot.png", "Spatial_Heatmap_of_UMI_Count_raw.png",
                         "Spatial_Heatmap_of_UMI_Count_log.png", "Threshold.png")
                         for (f in png_files) {
                           fileConn <- file(file.path(temp_dir, f))
                             writeLines("dummy image content", fileConn)
                               close(fileConn)}
html_files <- c("tsne_interactive.html", "umap_interactive.html")
for (f in html_files) {
 fileConn <- file(file.path(temp_dir, f))
 writeLines("<html>dummy interactive html</html>", fileConn)
 close(fileConn)}
dummy_rmd <- paste(
 "Report for stPipe",
 "Mapping Statistics: <<Mapping_statistics_plot>>",
 "UMI Duplication: <<UMI_duplication_plot>>",
 "Barcode Demultiplexing: <<Barcode_demultiplexing_plot>>",
 "Spatial Heatmap Raw: <<Spatial_Heatmap_of_UMI_Count_raw>>",
 "Spatial Heatmap Log: <<Spatial_Heatmap_of_UMI_Count_log>>",
 "Threshold: <<Threshold>>",
 "tSNE Interactive: <<tsne_interactive>>",
 "UMAP Interactive: <<umap_interactive>>",
 sep = "\n"
)
dummy_template_path <- file.path(temp_dir, "stPipe_Report_Skeleton.Rmd")
writeLines(dummy_rmd, dummy_template_path)
Run_HTML_test <- function(path) {
 rmd_template_path <- dummy_template_path
 if (rmd_template_path == "") {
  stop("Template file not found. Please check if it's properly installed in your package.")
 }
rmd_template <- readLines(rmd_template_path)
 gene_count <- read.csv(file.path(path, "gene_count.csv"), row.names = 1)
  Mapping_statistics_plot <- normalizePath(file.path(path, "Mapping_statistics_plot.png"))
 UMI_duplication_plot <- normalizePath(file.path(path, "UMI_duplication_plot.png"))
 Barcode_demultiplexing_plot <- normalizePath(file.path(path, "Barcode_demultiplexing_plot.png"))
 Spatial_Heatmap_of_UMI_Count_raw <- normalizePath(file.path(path, "Spatial_Heatmap_of_UMI_Count_raw.png"))
 Spatial_Heatmap_of_UMI_Count_log <- normalizePath(file.path(path, "Spatial_Heatmap_of_UMI_Count_log.png"))
 Threshold <- normalizePath(file.path(path, "Threshold.png"))
 tsne_interactive <- normalizePath(file.path(path, "tsne_interactive.html"))
 umap_interactive <- normalizePath(file.path(path, "umap_interactive.html"))
 rmd_content <- gsub("<<path>>", path, rmd_template)
 rmd_content <- gsub("<<Mapping_statistics_plot>>", Mapping_statistics_plot, rmd_content)
 rmd_content <- gsub("<<UMI_duplication_plot>>", UMI_duplication_plot, rmd_content)
 rmd_content <- gsub("<<Barcode_demultiplexing_plot>>", Barcode_demultiplexing_plot, rmd_content)
 rmd_content <- gsub("<<Spatial_Heatmap_of_UMI_Count_raw>>", Spatial_Heatmap_of_UMI_Count_raw, rmd_content)
```

```
 rmd_content <- gsub("<<Spatial_Heatmap_of_UMI_Count_log>>", Spatial_Heatmap_of_UMI_Count_log, rmd_content)
 rmd_content <- gsub("<<Threshold>>", Threshold, rmd_content)
 rmd_content <- gsub("<<tsne_interactive>>", tsne_interactive, rmd_content)
 rmd_content <- gsub("<<umap_interactive>>", umap_interactive, rmd_content)
 rmd_file <- file.path(path, "report.Rmd")
 writeLines(rmd_content, rmd_file)
 rmarkdown::render(rmd_file, output_file = file.path(path, "report.html"))}
Run_HTML_test(temp_dir)
```

`Run_Interactive` *Interactive Visualization for Spatial Transcriptomics Data and spot plot with save functionality*

Description

Interactive Visualization for Spatial Transcriptomics Data and spot plot with save functionality

Usage

```
Run_Interactive(
  matched_data,
  clustering_result,
  background_img = NULL,
  reduction_method = "tsne",
  point_size = 1
)
```

Arguments

`matched_data`	A data frame containing matched spatial coordinates with raw UMI counts.
`clustering_result`	A data frame containing matched spatial coordinates with raw UMI counts.
`background_img`	Optional background H&E image.
`reduction_method`	T-SNE ("tsne") or UMAP ("umap") result data frame obtained from 'Run_Clustering' function. Default set as "tsne".
`point_size`	Size of point shown in the spatial heatmap. Default set as 1.

Details

This function generates interactive plots to visualize spatial transcriptomics data. It takes matched spatial coordinates and raw UMI counts to produce customized t-SNE or UMAP plots overlayed on a optional background H&E image.

Value

R-shiny interactive webpage.

Examples

```
matched_data <- data.frame(
X_coordinate = runif(10, 0, 100),
Y_coordinate = runif(10, 0, 100),
UMI_count = sample(seq_len(100), 10),
spatial_name = paste0("Spot", seq_len(10)),
stringsAsFactors = FALSE
)
clustering_result <- data.frame(
TSNE1 = runif(10, -50, 50),
TSNE2 = runif(10, -50, 50),
spot = paste0("Spot", seq_len(10)),
cluster = sample(seq_len(3), 10, replace = TRUE),
stringsAsFactors = FALSE
)
if (interactive()) {
Run_Interactive(
  matched_data = matched_data,
  clustering_result = clustering_result,
  background_img = NULL,
  reduction_method = "tsne",
  point_size = 1)
                              }
```

Run_Loc_Match *Match spatial location After Pre-Processing for Sequencing-Based Spatial Transcriptomics*

Description

This function matches spatial coordinates for sST data after upstream pre-processing. 'Run_loc_match' can either map the technology coordination system (such as spot in Visium coordination, bead in Slide-seq coordination) or compute pixel for each spot and map the pixel information back to the image (only for Visium)

Usage

```
Run_Loc_Match(config, pixel = FALSE, show.config = TRUE)
```

Arguments

config	Path to the YAML configuration file.
pixel	Computing spot pixel or not. If yes, compute pixel for each spot and map back to image; if not, map the Visium coordination system. Defaults to FALSE.
show.config	Logical value indicating whether to print the configuration. Defaults to TRUE.

Value

A data frame contains gene count matrix with spatial coordinates

Examples

```
data_dir <- tempdir()
output_dir <- file.path(tempdir(), "Run_Loc_Match_output")
if (!dir.exists(output_dir)) dir.create(output_dir, recursive = TRUE)
sample_index <- data.frame(
  barcode_sequence = c("BC001", "BC002", "BC003"),
  cell_name = c("CELL_1", "CELL_2", "CELL_3"),
  stringsAsFactors = FALSE
)
write.csv(sample_index, file = file.path(output_dir, "sample_index.csv"), row.names = FALSE)
gene_count <- data.frame(
  gene_id = c("gene1", "gene2"),
  CELL_1 = c(10, 5),
  CELL_2 = c(20, 10),
  CELL_3 = c(30, 15),
  stringsAsFactors = FALSE
)
write.csv(gene_count, file = file.path(output_dir, "gene_count.csv"), row.names = FALSE)
config_list <- list(
  output_directory = output_dir,
  data_directory = data_dir,
  technology_version = "Visium_probe_v1",
  visium_coordination = "V1"
)
config_file <- tempfile(fileext = ".yml")
yaml::write_yaml(config_list, config_file)
result <- Run_Loc_Match(
  config = config_file,
  pixel = FALSE,
  show.config = FALSE
)
```

`Run_QC` *QC Control After Upstream Pre-Processing for Sequencing-Based Spatial Transcriptomics*

Description

QC Control After Upstream Pre-Processing for Sequencing-Based Spatial Transcriptomics

Usage

```
Run_QC(config, matched.data, gene.matrix, show.config = TRUE)
```

Arguments

`config`	Path to the YAML configuration file.
`matched.data`	A data frame containing spatial transcriptomics data, including UMI counts and spatial coordinates, this is usually obtained from 'Run_loc_match' function.

`gene.matrix`	A gene count matrix, this is usually obtained from 'Run_ST' function.
`show.config`	Logical value indicating whether to print the configuration. Defaults to TRUE.

Details

This function performs QC control on sequencing-based spatial transcriptomics data after upstream pre-processing step such as 'Run_ST' step. Ensure the output directory is the same with the 'Run_ST' one. Filtering is performed either use specific UMI threshold or assign the threshold to 'DropletUtils'.

"max_slope"

In this approach, the filtering is done based on UMI counts. Spots with counts below a certain threshold are considered low-quality and are filtered out. This method helps retain only the spots with significant transcriptomic signals, reducing noise from sptos with minimal or no meaningful biological information.

Threshold Determination: The threshold in this method is computed by analyzing the distribution of UMI counts across spots, and identifying the point of maximum slope in the cumulative UMI distribution curve. This point often corresponds to the transition between background noise and real biological signals.

"EmptyDropletUtils"

Alternatively, the `DropletUtils` package offers a more sophisticated approach by using statistical methods to identify droplets or spots that contain real cells, as opposed to empty droplets or those containing background RNA. This method calculates a false discovery rate (FDR) to assess the likelihood of each droplet containing a real cell. Sptos are retained if they meet the significance criteria for either the p-value or FDR. To learn more details regarding `DropletUtils`, visit this link to its Bioconductor page.

Multiple Thresholds: This method will determine two thresholds based on the config file input parameters. Filtering can be fine-tuned using both p-value and FDR thresholds, offering greater flexibility in distinguishing between noise and meaningful data. Sptos are retained if they meet the significance criteria for either the p-value or FDR.

Value

A list containing filtered gene counts with matched spatial coordinates after QC.

Examples

```
output_dir <- tempdir()
config_list <- list(
output_directory = output_dir,
qc_filter = "slope_max",
qc_per = "0.4_0.8"
)
config_path <- tempfile(fileext = ".yml")
yaml::write_yaml(config_list, config_path)
set.seed(123)
gene_ids <- paste0("gene", seq_len(100))
spatial_names <- paste0("SPATIAL_", seq_len(100))
count_matrix <- matrix(rpois(100*100, lambda = 20), nrow = 100, ncol = 100)
```

```
colnames(count_matrix) <- spatial_names
gene_matrix <- data.frame(row.names = gene_ids, count_matrix, stringsAsFactors = FALSE)
matched_data <- data.frame(
 X_coordinate = runif(100, min = 0, max = 1000),
 Y_coordinate = runif(100, min = 0, max = 1000),
 barcode_sequence = paste0("BC", seq_len(100)),
 spatial_name = spatial_names,
 stringsAsFactors = FALSE
)
umi_counts <- colSums(gene_matrix[, -1])
matched_data$UMI_count <- umi_counts[match(matched_data$spatial_name, names(umi_counts))]
qc_results <- Run_QC(
 config = config_path,
 matched.data = matched_data,
 gene.matrix = gene_matrix,
 show.config = FALSE
)
```

Run_ST *Pre-processing function for sequencing-based spatial transcriptomics*

Description

Pre-processing from FASTQ files to gene count matrix and extract spatial location information, processing multiple sample is done via parallel computing. NB: for multiple samples, 'species' and 'technology_version' should be the same. This function processes sequencing-based spatial transcriptomics data using various steps, including BAM to FASTQ conversion, trimming, index building, alignment, and barcode detection. For Slideseq technology, the input should be BAM file and for all other technologies the input should be FASTQ file.

Usage

```
Run_ST(config, show.config = TRUE)
```

Arguments

config	Path to the YAML configuration file.
show.config	Logical value indicating whether to print the configuration. Defaults to TRUE.

Value

None. Outputs are saved to specified directories.

Examples

```
data_path <- system.file("extdata", package = "stPipe")
output_directory <- file.path(tempdir(), "stPipe_output")
config_list <- list(
data_directory = data_path,
```

```
output_directory = output_directory,
technology_version = "Visium_probe_v1",
species = "mouse",
scpipe_nthreads = 4,
max_reads = 100000,
min_count = 10,
number_of_locations = 100,
bs1= -1,
bl1= 0,
bs2= 0,
bl2= 16,
us= 16,
ul= 12,
ll= 0
)
config_file <- tempfile(fileext = ".yml")
yaml::write_yaml(config_list, config_file)
Run_ST(
  config = config_file,
  show.config = FALSE
)
```

`Run_Visualization` *Visualize pre-processed sST data*

Description

This function visualizes both spatial-level and read-level information either before or after the 'Run_QC' step. It outputs raw and log-transformed UMI count plots for spatial flag, and demultiplexing and mapping statistics for read flag.

Usage

```
Run_Visualization(
  matched.data = NULL,
  config,
  Vis.spatial = TRUE,
  Vis.read = TRUE,
  show.config = TRUE
)
```

Arguments

`matched.data`	A data frame containing spatial transcriptomics data, including UMI counts and spatial coordinates. This is usually obtained from 'Run_loc_match' function.
`config`	Path to the YAML configuration file.
`Vis.spatial`	Logical value indicating whether to visualize spatial data. Defaults to TRUE.
`Vis.read`	Logical value indicating whether to visualize read-level data. Defaults to TRUE.
`show.config`	Logical value indicating whether to print the configuration. Defaults to TRUE.

Value

A list contains spatial and read level visualization results.

Examples

```
temp_config <- tempfile(fileext = ".yml")
writeLines("technology_version: 'Visium 1.0'\noutput_directory: '.'", temp_config)
matched.data <- data.frame(
X_coordinate = runif(10, 0, 100),
Y_coordinate = runif(10, 0, 100),
UMI_count = sample(seq_len(100), 10),
spatial_name = paste0("Spot", seq_len(10)),
stringsAsFactors = FALSE
)
vis_results <- Run_Visualization(
  matched.data = matched.data,
  config = temp_config,
  Vis.spatial = TRUE,
  Vis.read = FALSE,
  show.config = FALSE
)
```

---

stPipe *stPipe: A package for pre-processing sequencing-based spatial transcriptomics data.*

Description

The stPipe will do spatial barcode demultiplexing, UMI deduplication, spatial location matching and quality control on fastq data generated from all mainstream protocols

stPipe functions

The stPipe functions Run_ST, Run_loc_match, Run_QC, Run_Vis, Run_Clustering, Run_create_obj, Run_HTML, Run_Interactive

Author(s)

Yang Xu <xu.ya@wehi.edu.au>

See Also

Useful links:

- https://github.com/mritchielab/stPipe
- Report bugs at https://github.com/mritchielab/stPipe/issues/new

Index

Run_Clustering, 2
Run_Create_Obj, 3
Run_HTML, 4
Run_Interactive, 6
Run_Loc_Match, 7
Run_QC, 8
Run_ST, 10
Run_Visualization, 11

stPipe, 12
stPipe-package (stPipe), 12