Package: sciNOME (via r-universe)

June 14, 2026

Title Region-Centric Integration for Single-Cell Multi-Omics

Version 0.99.0

Description Provides an end-to-end pipeline for processing RNA, DNA methylation, and chromatin accessibility data at the single-cell level. It features ultra-fast data aggregation, quality control, dimensionality reduction, and multi-omics integration based on genomic regions. Additionally, it offers extensive publication-ready visualization tools for downstream analysis.

License GPL-3

URL https://github.com/Medinfo-Lab/sciNOME

BugReports https://github.com/Medinfo-Lab/sciNOME/issues

biocViews Software, SingleCell, Epigenetics, Transcriptomics, DNAMethylation, DifferentialExpression, DimensionReduction, Clustering, QualityControl

Depends R (>= 4.2.0)

Imports data.table, dplyr, FNN, GenomicRanges, ggplot2, grDevices, ggpubr, ggrepel, ggridges, igraph, impute, IRanges, irlba, magrittr, Matrix, methods, NMF, parallel, patchwork, pbapply, princurve, Rtsne, S4Vectors, stats, tidyr, tidyselect, tools, utils, uwot

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

VignetteBuilder knitr

Encoding UTF-8

LazyData false

RoxygenNote 7.3.2

Config/testthat/edition 3

Config/pak/sysreqs cmake libglpk-dev make libicu-dev libxml2-dev

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

Date/Publication 2026-06-14 04:00:54 UTC

RemoteUrl https://github.com/BiocStaging/sciNOME

RemoteRef HEAD

RemoteSha 890d8da44f95cacec589dd82a9b6f3e4e85b51ca

Contents

AddMetaData_RNA . . . 2
Aggregate_epiRegions . . . 3
Build_RNAObject . . . 4
CreateRNAObject . . . 6
Extract_epiData . . . 7
Integrate_MultiOmics . . . 8
PlotDimRed_Epi . . . 10
PlotDimRed_RNA . . . 11
PlotLandscape_Epi . . . 12
PlotOmicsMatrix . . . 13
PlotOmicsScatter . . . 14
PlotPseudo_RNA . . . 16
PlotQC_RNA . . . 17
PlotVolcano_Epi . . . 18
PlotVolcano_RNA . . . 19
ProcessQC_RNA . . . 20
QC_epiData . . . 21
Reduce_epiDims . . . 23
Run_Diffanalysis . . . 24
RunClustering_RNA . . . 26
RunDEA_RNA . . . 27
RunDimReduction_RNA . . . 28
RunPseudotime_RNA . . . 30

Index **32**

`AddMetaData_RNA` *Add Metadata to a RNA Object*

Description

Core engine for injecting metadata into the `meta.data` slot of a RNA object. Supports both strict ID matching and ordered vector injection.

Usage

```
AddMetaData_RNA(object, meta_data, id_col = NULL, group_cols = NULL)
```

Arguments

object	A RNA object.
meta_data	A data.frame containing metadata to add.
id_col	Character. Column name in meta_data to match with cell names. If NULL or "MATCH_BY_ORDER", matches by row order.
group_cols	Character vector of column names to add. If NULL, adds all columns except id_col.

Value

An updated RNA object with integrated metadata.

Examples

```
# 1. Create a basic RNA object
mock_counts <- matrix(rpois(100, lambda = 5), nrow = 10, ncol = 10)
rownames(mock_counts) <- paste0("Gene_", 1:10)
colnames(mock_counts) <- paste0("Cell_", 1:10)
rna_obj <- CreateRNAObject(counts = mock_counts)

# 2. Create mock metadata
meta_data <- data.frame(
  Cell_ID = paste0("Cell_", 1:10),
  Group = rep(c("Control", "Treatment"), each = 5),
  Batch = rep(c("B1", "B2"), times = 5)
)

# 3. Add metadata to the object
rna_obj <- AddMetaData_RNA(rna_obj, meta_data = meta_data, id_col = "Cell_ID")
head(rna_obj$meta.data)
```

Aggregate_epiRegions *Aggregate Bismark Coverage Files into Defined Genomic Regions*

Description

This function takes a directory of Bismark '.cov.gz' files and a BED file containing genomic regions. It aggregates the methylation data across these regions in parallel.

Usage

```
Aggregate_epiRegions(cov_dir, bed_file, n_cores = 1)
```

Arguments

cov_dir	Character. Path to the directory containing '.cov.gz' files.
bed_file	Character. Path to the BED file. Must contain at least 3 columns: chr, start, end.
n_cores	Integer. Number of cores to use for parallel processing. Default is 1.

Value

A `data.table` containing the aggregated results. The first column is `region_id`, followed by `.meth`, `.nonmeth`, and `.level` columns for each sample.

Examples

```
# 1. Create a temporary directory for mock files
tmp_dir <- tempdir()

# 2. Create a mock BED file
mock_bed <- data.frame(chr = "chr1", start = 1, end = 100)
bed_path <- file.path(tmp_dir, "mock.bed")
write.table(mock_bed, bed_path, row.names=FALSE, col.names=FALSE, sep="\t", quote=FALSE)

# 3. Create a mock Bismark coverage file (.cov)
mock_cov <- data.frame(
  chr = c("chr1", "chr1"),
  start = c(10, 50),
  end = c(10, 50),
  meth_perc = c(100, 0),
  meth = c(1, 0),
  unmeth = c(0, 1)
)
cov_path <- file.path(tmp_dir, "Sample1.cov")
write.table(mock_cov, cov_path, row.names=FALSE, col.names=FALSE, sep="\t", quote=FALSE)

# 4. Run the aggregation function
result <- Aggregate_epiRegions(cov_dir = tmp_dir, bed_file = bed_path, n_cores = 1)
head(result)
```

Build_RNAObject *Build a Complete RNA Object*

Description

One-step function to generate a RNA object directly from an expression matrix and a metadata data.frame. Automatically handles data cleaning, NA replacement, pre-filtering, and QC metrics calculation.

Usage

```
Build_RNAObject(
  expr_mat,
  meta_data = NULL,
  meta_id_col = NULL,
  meta_group_cols = NULL,
  min_cells = 0,
  min_features = 0,
  mt_pattern = "^[Mm][Tt]-",
```

```
  project_name = "RNA_Project"
)
```

Arguments

expr_mat	Expression matrix (data.frame or matrix), rows are genes, columns are samples.
meta_data	Metadata data.frame for sample grouping.
meta_id_col	Character. Column in meta_data matching matrix column names. Defaults to ordered matching if NULL.
meta_group_cols	Character vector. Specific metadata columns to add (Defaults to NULL, adding all columns).
min_cells	Integer. Pre-filtering: keep genes expressed in at least this many cells.
min_features	Integer. Pre-filtering: keep cells expressing at least this many genes.
mt_pattern	Character string. Regex pattern for mitochondrial genes. Default is "^[Mm][Tt]-".
project_name	Character. Project name.

Value

A built and pre-filtered RNA object.

Examples

```
# 1. Create mock expression matrix and metadata
set.seed(123)
mock_counts <- matrix(rpois(100, lambda = 10), nrow = 10, ncol = 10)
rownames(mock_counts) <- paste0("Gene_", 1:10)
colnames(mock_counts) <- paste0("Cell_", 1:10)

mock_meta <- data.frame(
  ID = paste0("Cell_", 1:10),
  Condition = rep(c("WT", "KO"), each = 5)
)

# 2. Build the object directly
rna_obj <- Build_RNAObject(
  expr_mat = mock_counts,
  meta_data = mock_meta,
  meta_id_col = "ID",
  min_cells = 0, # Set to 0 for this tiny mock data
  min_features = 0
)
head(rna_obj$meta.data)
```

`CreateRNAObject` *Create a Lightweight RNA Object*

Description

Internal core function to initialize a RNA object from a raw counts matrix. Automatically calculates basic QC metrics including percent_mt.

Usage

```
CreateRNAObject(
  counts,
  project_name = "RNA_Project",
  mt_pattern = "^[Mm][Tt]-"
)
```

Arguments

`counts`	A sparse matrix or data.frame containing raw expression counts.
`project_name`	Character string specifying the project name. Default is "RNA_Project".
`mt_pattern`	Character string. Regex pattern to identify mitochondrial genes. Default is "^[Mm][Tt]-".

Value

A `RNA` object containing initialized assays and basic QC metrics.

Examples

```
# Create a mock count matrix (10 genes x 10 cells)
set.seed(42)
mock_counts <- matrix(rpois(100, lambda = 5), nrow = 10, ncol = 10)
rownames(mock_counts) <- paste0("Gene_", 1:10)
colnames(mock_counts) <- paste0("Cell_", 1:10)

# Initialize the RNA object
rna_obj <- CreateRNAObject(counts = mock_counts, project_name = "Mock_Project")
print(class(rna_obj))
head(rna_obj$meta.data)
```

`Extract_epiData` *Extract Specific Metric Matrix from Aggregated Epigenetic Data*

Description

Extracts columns ending with a specific suffix (e.g., ".meth", ".nonmeth", ".level") from the aggregated results, while always retaining the 'region_id' column.

Usage

```
Extract_epiData(data, suffix, clean_names = TRUE)
```

Arguments

`data`	A `data.frame` or `data.table` generated by `aggregate_epi_regions`.
`suffix`	Character. The suffix to extract. Examples: ".meth", ".nonmeth", ".level".
`clean_names`	Logical. If `TRUE` (default), removes the suffix from the column names in the returned data, leaving only the clean sample names.

Value

A `data.table` containing 'region_id' and the extracted metric columns.

Examples

```
# Create mock aggregated data
mock_data <- data.frame(
  region_id = c("chr1:1-100", "chr1:101-200"),
  SampleA.meth = c(10, 20),
  SampleA.level = c(0.8, 0.9),
  SampleB.meth = c(15, 25),
  SampleB.level = c(0.5, 0.6)
)

# Extract only the '.level' columns
level_data <- Extract_epiData(data = mock_data, suffix = ".level")
head(level_data)
```

`Integrate_MultiOmics` *Integrate RNA, DNA Methylation (CpG), and Chromatin Accessibility (GpC) Data*

Description

Merges multiple omics layers based on genomic region annotations and Gene IDs. Uses a single global metadata table to safely map samples across different omics matrices. Optionally filters features based on differential analysis results.

Usage

```
Integrate_MultiOmics(
  mode = c("tri", "rna_cpg", "rna_gpc", "cpg_gpc"),
  target_group,
  meta_df,
  group_col = "group1",
  region_df,
  rna_obj = NULL,
  rna_id_col = "sample",
  rna_diff_df = NULL,
  rna_pval_col = "p_val",
  rna_logfc_col = "avg_log2FC",
  rna_pval_th = 0.05,
  rna_logfc_th = 0.5,
  cpg_mat = NULL,
  cpg_id_col = "CpG_level",
  cpg_diff_df = NULL,
  cpg_pval_col = "P.Value",
  cpg_diff_col = "Diff",
  cpg_pval_th = 0.05,
  cpg_diff_th = 0.1,
  gpc_mat = NULL,
  gpc_id_col = "GpC_level",
  gpc_diff_df = NULL,
  gpc_pval_col = "P.Value",
  gpc_diff_col = "Diff",
  gpc_pval_th = 0.05,
  gpc_diff_th = 0.1
)
```

Arguments

`mode`	Character. Integration mode: "tri" (all 3), "rna_cpg", "rna_gpc", or "cpg_gpc".
`target_group`	Character. The biological group/condition to calculate averages for.
`meta_df`	Data.frame. The global metadata linking all omics samples (e.g., "ALL" sheet).

group_col	Character. The column name in meta_df containing group info (e.g., "group1").
region_df	Data.frame. Genomic region annotations (must contain chr, start, end, and gene id).
rna_obj	Object. The scRNA-seq object.
rna_id_col	Character. Column in meta_df matching RNA cell/sample names.
rna_diff_df	Data.frame. Optional. RNA differential analysis results.
rna_pval_col	Character. P-value column in rna_diff_df.
rna_logfc_col	Character. logFC column in rna_diff_df.
rna_pval_th	Numeric. P-value threshold for RNA.
rna_logfc_th	Numeric. Absolute logFC threshold for RNA.
cpg_mat	Matrix/Data.frame. CpG level matrix (Rows = Regions, Cols = Samples).
cpg_id_col	Character. Column in meta_df matching CpG matrix columns.
cpg_diff_df	Data.frame. Optional. CpG DMR analysis results.
cpg_pval_col	Character. P-value column in cpg_diff_df.
cpg_diff_col	Character. Difference column in cpg_diff_df.
cpg_pval_th	Numeric. P-value threshold for CpG.
cpg_diff_th	Numeric. Absolute difference threshold for CpG.
gpc_mat	Matrix/Data.frame. GpC level matrix.
gpc_id_col	Character. Column in meta_df matching GpC matrix columns.
gpc_diff_df	Data.frame. Optional. GpC DMR analysis results.
gpc_pval_col	Character. P-value column in gpc_diff_df.
gpc_diff_col	Character. Difference column in gpc_diff_df.
gpc_pval_th	Numeric. P-value threshold for GpC.
gpc_diff_th	Numeric. Absolute difference threshold for GpC.

Value

A merged data.frame containing integrated omics levels.

Examples

```
# 1. Create mock metadata
meta_df <- data.frame(
  group1 = c("Tumor", "Tumor", "Normal"),
  sample = c("cell1", "cell2", "cell3"),
  CpG_level = c("samp1", "samp2", "samp3")
)

# 2. Create mock region annotation
region_df <- data.frame(
  chr = c("chr1", "chr2"),
  start = c(100, 200),
  end = c(150, 250),
```

```
  gene_id = c("GeneA", "GeneB")
)

# 3. Create mock RNA object
rna_mat <- matrix(
  c(10, 12, 2, 5, 6, 1), nrow = 2, byrow = TRUE,
  dimnames = list(c("GeneA", "GeneB"), c("cell1", "cell2", "cell3"))
)
rna_obj <- list(assays = list(RNA = list(data = rna_mat)))

# 4. Create mock CpG matrix (rownames match chr:start-end from region_df)
cpg_mat <- matrix(
  c(0.8, 0.9, 0.1, 0.7, 0.8, 0.2), nrow = 2, byrow = TRUE,
  dimnames = list(c("chr1:100-150", "chr2:200-250"), c("samp1", "samp2", "samp3"))
)

# 5. Run Integration
merged_res <- Integrate_MultiOmics(
  mode = "rna_cpg",
  target_group = "Tumor",
  meta_df = meta_df,
  group_col = "group1",
  region_df = region_df,
  rna_obj = rna_obj, rna_id_col = "sample",
  cpg_mat = cpg_mat, cpg_id_col = "CpG_level"
)

head(merged_res)
```

`PlotDimRed_Epi` *Plot Dimensionality Reduction Results*

Description

Generates an elegant, publication-ready scatter plot from the output of ‘Reduce_epiDims‘. Features dynamic coloring, automatic confidence ellipses (if sample size allows), and clean themes.

Usage

```
PlotDimRed_Epi(dr_data, group_col, method = "PCA", title = NULL)
```

Arguments

`dr_data`	Data.frame. The output from ‘Reduce_epiDims‘ (must contain Dim1, Dim2, and the grouping column).
`group_col`	Character. The name of the column in ‘dr_data‘ used for grouping/coloring.
`method`	Character. The dimension reduction method used (e.g., "PCA", "UMAP"). Used for axis labels and title.
`title`	Character. Optional custom title. If NULL, defaults to the ‘method‘ name.

Value

A ggplot2 object.

Examples

```
# 1. Create mock dimensionality reduction output data
set.seed(123)
mock_dr <- data.frame(
  Dim1 = rnorm(15),
  Dim2 = rnorm(15),
  Group = rep(c("Tumor", "Normal", "Control"), each = 5)
)

# 2. Plot PCA with confidence ellipses
p_pca <- PlotDimRed_Epi(
  dr_data = mock_dr,
  group_col = "Group",
  method = "PCA"
)
p_pca
```

PlotDimRed_RNA *Plot Dimensionality Reduction (PCA, t-SNE, UMAP)*

Description

Generates scatter plots for dimensionality reduction coordinates. Supports coloring by custom metadata and optionally juxtaposing with unsupervised clustering results. Uses distinct color palettes for the metadata plot and the clustering plot to avoid visual confusion.

Usage

```
PlotDimRed_RNA(
  obj,
  reduction = "umap",
  group_col = "orig.ident",
  show_cluster = FALSE
)
```

Arguments

obj	A RNA object containing reductions and metadata.
reduction	Character. The reduction to plot: "umap", "tsne", or "pca".
group_col	Character. The column name in metadata to color the points by (e.g., "Condition", "Batch").
show_cluster	Logical. If TRUE, plots a side-by-side comparison with 'Auto_Cluster' (requires patchwork package).

Value

A ggplot or patchwork object containing the generated plot(s).

Examples

```
# 1. Create minimal mock RNA object with reductions and metadata
set.seed(42)
mock_umap <- matrix(rnorm(80), ncol = 2)
rownames(mock_umap) <- paste0("Cell_", 1:40)
colnames(mock_umap) <- c("UMAP_1", "UMAP_2")

mock_meta <- data.frame(
  orig.ident = rep(c("Control", "Treatment"), each = 20),
  Auto_Cluster = rep(paste0("Cluster_", 1:4), times = 10)
)
rownames(mock_meta) <- rownames(mock_umap)

mock_obj <- list(
  reductions = list(umap = mock_umap),
  meta.data = mock_meta
)
class(mock_obj) <- "RNA"

# 2. Generate plot
p_dim <- PlotDimRed_RNA(mock_obj, reduction = "umap", group_col = "orig.ident")
p_dim
```

PlotLandscape_Epi *Plot Epigenetic Landscape (Ridge Plot) from Level Matrix*

Description

Calculates the global mean across features for each sample from a provided '.level' matrix and generates a high-quality ridge plot showing the distribution across specified categorical groups.

Usage

```
PlotLandscape_Epi(
  mat,
  meta,
  sample_col,
  group_col,
  title = "Global Epigenetic Landscape",
  subtitle = "Distribution of Mean Epigenetic Levels per Single Cell",
  xlab = "Global Mean Level",
  ylab = "Group / Condition"
)
```

Arguments

mat	A numeric matrix or data.frame of extraction levels (Rows = Features/Regions, Columns = Samples).
meta	A data.frame containing metadata/sample information.
sample_col	Character. The column name in meta containing sample IDs matching the colnames of mat.
group_col	Character. The column name in meta representing the groups/conditions for the Y-axis.
title	Character. The main title of the plot.
subtitle	Character. The subtitle of the plot.
xlab	Character. The X-axis label.
ylab	Character. The Y-axis label.

Value

A ggplot object.

Examples

```
# 1. Create a mock epigenetic level matrix (Features x Samples)
set.seed(42)
mock_mat <- matrix(runif(150, min = 0, max = 1), nrow = 10, ncol = 15)
colnames(mock_mat) <- paste0("Sample_", 1:15)
rownames(mock_mat) <- paste0("Region_", 1:10)

# 2. Create corresponding metadata
mock_meta <- data.frame(
  SampleID = paste0("Sample_", 1:15),
  Condition = rep(c("Tumor", "Normal", "Metastasis"), each = 5)
)

# 3. Generate Ridge Plot (Requires ggridges)
if (requireNamespace("ggridges", quietly = TRUE)) {
  p_ridge <- PlotLandscape_Epi(
    mat = mock_mat, meta = mock_meta,
    sample_col = "SampleID", group_col = "Condition"
  )
  p_ridge
}
```

PlotOmicsMatrix *Plot Multi-Omics Correlation Matrix*

Description

Generates a matrix plot showing the distribution (density) of each omics layer on the diagonal, pairwise scatter plots on the lower triangle, and Spearman correlation coefficients on the upper triangle. Automatically adapts to dual-omics (2x2) or tri-omics (3x3) data based on the input dataframe.

Usage

```
PlotOmicsMatrix(
  df,
  title = "Omics Correlation Matrix",
  cor_method = "spearman",
  point_color = "#3C5488",
  density_fill = "#B09C85",
  alpha = 0.5
)
```

Arguments

df	Data.frame. The integrated multi-omics dataframe output by IntegrateMultiOmics.
title	Character. Global title for the plot.
cor_method	Character. Correlation method to use ("spearman", "pearson", or "kendall").
point_color	Character. Color of the scatter plot points.
density_fill	Character. Fill color for the density plots.
alpha	Numeric. Transparency for points and density fills (0 to 1).

Value

A ggplot object representing the correlation plots.

Examples

```
# 1. Create mock multi-omics integrated dataframe
set.seed(123)
mock_omics <- data.frame(
  RNA_Exp = rnorm(50, mean = 10, sd = 2),
  CpG_level = runif(50, 0, 1),
  GpC_level = runif(50, 0, 1)
)

# 2. Generate Omics Correlation Matrix Plot (requires patchwork)
if (requireNamespace("patchwork", quietly = TRUE)) {
  p_matrix <- PlotOmicsMatrix(mock_omics, cor_method = "spearman")
  p_matrix
}
```

PlotOmicsScatter *Plot Multi-Omics Pairwise Scatter Plots*

Description

Generates pairwise scatter plots for integrated multi-omics data. Automatically adds linear regression trend lines (LM) and calculates correlation coefficients (Spearman by default) using ggpubr::stat_cor. Dynamically generates 1 to 3 plots depending on the available omics layers in the dataframe.

Usage

```
PlotOmicsScatter(
  df,
  cor_method = "spearman",
  point_size = 1.5,
  point_alpha = 0.6,
  color_cpg_rna = "#4DBBD5",
  color_gpc_rna = "#00A087",
  color_cpg_gpc = "#E64B35"
)
```

Arguments

df	Data.frame. The integrated multi-omics dataframe output by IntegrateMultiOmics.
cor_method	Character. Correlation method: "spearman" (default) or "pearson".
point_size	Numeric. Size of the scatter points.
point_alpha	Numeric. Transparency of the scatter points (0 to 1).
color_cpg_rna	Character. Color for CpG vs RNA plot.
color_gpc_rna	Character. Color for GpC vs RNA plot.
color_cpg_gpc	Character. Color for CpG vs GpC plot.

Value

A ggplot object representing the scatter plots.

Examples

```
# 1. Create mock multi-omics integrated dataframe
set.seed(42)
mock_omics <- data.frame(
  RNA_Exp = rnorm(30, mean = 5, sd = 1),
  CpG_level = runif(30, 0, 1)
  # Omitted GpC_level to test the 2-omics dynamic layout
)

# Add artificial correlation
mock_omics$RNA_Exp <- mock_omics$RNA_Exp - (mock_omics$CpG_level * 2)

# 2. Generate pairwise scatter plots (requires ggpubr and patchwork)
if (requireNamespace("ggpubr", quietly = TRUE) &&
    requireNamespace("patchwork", quietly = TRUE)) {
  p_scatter <- PlotOmicsScatter(mock_omics)
  p_scatter
}
```

PlotPseudo_RNA *Plot Trajectory and Pseudotime Inference*

Description

Generates a dual-panel plot for pseudotime analysis. The left panel shows the continuous pseudotime gradient, while the right panel shows the categorical clusters. Overlays the inferred trajectory path and the root node (start cluster) if available.

Usage

```
PlotPseudo_RNA(obj)
```

Arguments

obj	A RNA object containing pseudotime results in reductions$pseudotime.

Value

A patchwork object containing the combined trajectory plots.

Examples

```
# 1. Create mock RNA object with trajectory data
set.seed(123)
mock_umap <- matrix(rnorm(40), ncol = 2)
rownames(mock_umap) <- paste0("Cell_", 1:20)
colnames(mock_umap) <- c("UMAP_1", "UMAP_2")

mock_meta <- data.frame(Auto_Cluster = rep(c("Cluster_1", "Cluster_2"), each = 10))
rownames(mock_meta) <- rownames(mock_umap)

# Mock pseudotime structure
mock_pseudotime <- list(
  pseudotime = runif(20, 0, 1),
  dr_method = "umap",
  group_col = "Auto_Cluster",
  algorithm = "cluster",
  start_clus = "Cluster_1",
  curve_coords = matrix(rnorm(40), ncol = 2)
)

mock_obj <- list(
  reductions = list(umap = mock_umap, pseudotime = mock_pseudotime),
  meta.data = mock_meta
)
class(mock_obj) <- "RNA"

# 2. Plot trajectory (requires patchwork)
if (requireNamespace("patchwork", quietly = TRUE)) {
```

```
  p_pseudo <- PlotPseudo_RNA(mock_obj)
  p_pseudo
}
```

PlotQC_RNA *Plot Quality Control Metrics*

Description

Generates a grouped violin plot for quality control metrics (e.g., nFeature_RNA, nCount_RNA, percent_mt) across specified cell groups. Automatically handles color interpolation for large numbers of groups.

Usage

```
PlotQC_RNA(
  obj,
  group_col = NULL,
  features = c("nFeature_RNA", "nCount_RNA", "percent_mt")
)
```

Arguments

obj	A RNA object containing calculated QC metrics in meta.data.
group_col	Character. The column name in meta.data to group cells by. If NULL or not found, defaults to "orig.ident".
features	Character vector. The QC metrics to plot. Defaults to c("nFeature_RNA", "nCount_RNA", "percent_mt").

Value

A ggplot object representing the QC metrics violin plots.

Examples

```
# 1. Create a minimal mock RNA object with required meta.data
mock_meta <- data.frame(
  orig.ident = rep(c("Sample_A", "Sample_B"), each = 20),
  nFeature_RNA = runif(40, min = 200, max = 2500),
  nCount_RNA = runif(40, min = 500, max = 8000),
  percent_mt = runif(40, min = 0, max = 15)
)
rownames(mock_meta) <- paste0("Cell_", 1:40)

mock_obj <- list(meta.data = mock_meta)
class(mock_obj) <- "RNA"

# 2. Generate QC plot
p_qc <- PlotQC_RNA(mock_obj, group_col = "orig.ident")
p_qc
```

PlotVolcano_Epi *Plot Volcano Plot for Differential Epigenetic Analysis*

Description

Generates a publication-quality volcano plot to visualize differential methylation/epigenetic regions (DMRs). Supports dynamic effect size metrics (e.g., logFC, Difference, Z-Scores).

Usage

```
PlotVolcano_Epi(
  df,
  metric_col = "Diff",
  p_col = "P.Value",
  feature_col = "chrdata",
  th_effect = 1,
  th_p = 0.05,
  top_n = 10,
  title = "Volcano Plot",
  custom_xlab = NULL
)
```

Arguments

df	A data.frame containing differential analysis results.
metric_col	Character. The column name for the effect size (X-axis). Default is "Diff".
p_col	Character. The column name for the P-values (Y-axis). Default is "P.Value".
feature_col	Character. The column name for feature names (labels). Default is "chrdata".
th_effect	Numeric. The absolute threshold for the effect size (e.g., logFC > 1). Default is 1.0.
th_p	Numeric. The threshold for statistical significance (e.g., P-value < 0.05). Default is 0.05.
top_n	Integer. Number of top significant features to label. Default is 10.
title	Character. The main title of the plot. Default is "Volcano Plot".
custom_xlab	Character or Expression. Custom label for the X-axis. If NULL, auto-generated based on metric_col.

Value

A ggplot object.

Examples

```
# 1. Create mock Differential Epigenetic Analysis (DEA) results
set.seed(42)
mock_epi_dea <- data.frame(
  chrdata = paste0("chr1:", 1000:(1000+99)),
  Diff = rnorm(100, mean = 0, sd = 0.8),
  P.Value = runif(100, min = 0, max = 0.5)
)
# Force some significant points
mock_epi_dea$Diff[1:3] <- c(1.5, 2.0, -1.8)
mock_epi_dea$P.Value[1:3] <- c(0.001, 0.0005, 0.002)

# 2. Plot Epigenetic Volcano Plot
p_vol <- PlotVolcano_Epi(
  df = mock_epi_dea,
  metric_col = "Diff", p_col = "P.Value", feature_col = "chrdata",
  th_effect = 1.0, th_p = 0.05, top_n = 3
)
p_vol
```

PlotVolcano_RNA *Plot Volcano Plot for Differential Expression Analysis*

Description

Generates an academic-grade volcano plot based on differential expression analysis results. Automatically highlights and labels top significant up- and down-regulated genes.

Usage

```
PlotVolcano_RNA(
  res_df,
  fc_cut = 1,
  p_cut = 0.05,
  top_n = 10,
  plot_title = NULL
)
```

Arguments

res_df	A data.frame containing DEA results. Must contain 'avg_log2FC' and 'p_val_adj' columns.
fc_cut	Numeric. The absolute log2 fold change threshold. Default is 0.5.
p_cut	Numeric. The adjusted p-value threshold. Default is 0.05.
top_n	Integer. The number of top up/down regulated genes to label. Default is 10.
plot_title	Character. Optional custom title. If NULL, auto-generates one.

Value

A ggplot object representing the volcano plot.

Examples

```
# 1. Generate mock Differential Expression Analysis (DEA) results
set.seed(123)
mock_dea <- data.frame(
  gene = paste0("Gene_", 1:100),
  avg_log2FC = rnorm(100, mean = 0, sd = 1.5),
  p_val_adj = runif(100, min = 0, max = 0.1)
)
# Make a few genes highly significant for the plot
mock_dea$avg_log2FC[1:5] <- runif(5, 2, 4)
mock_dea$p_val_adj[1:5] <- runif(5, 1e-10, 1e-5)

# 2. Generate Volcano Plot
p_volcano <- PlotVolcano_RNA(mock_dea, fc_cut = 1, p_cut = 0.05, top_n = 3)
p_volcano
```

ProcessQC_RNA *Process QC, Filtering, and Normalization for RNA Object*

Description

Performs mitochondrial proportion calculation, rigorous cell filtering, sparse matrix normalization (LogNormalize, LogCPM, or TPM), and optional data scaling.

Usage

```
ProcessQC_RNA(
  obj,
  mt_pattern = "^MT-",
  min_nCount = 500,
  max_nCount = 50000,
  min_nFeature = 200,
  max_nFeature = 8000,
  max_mt = 15,
  norm_method = "LogNormalize",
  do_scale = TRUE
)
```

Arguments

obj	A RNA object.
mt_pattern	Character. Regex pattern to identify mitochondrial genes (e.g., "^MT-" for human, "^mt-" for mouse).

min_nCount	Numeric. Minimum UMI count threshold.
max_nCount	Numeric. Maximum UMI count threshold.
min_nFeature	Numeric. Minimum number of detected genes.
max_nFeature	Numeric. Maximum number of detected genes.
max_mt	Numeric. Maximum allowed mitochondrial percentage (e.g., 15 means 15%).
norm_method	Character. Normalization method: "LogNormalize", "LogCPM", or "TPM".
do_scale	Logical. Whether to center and scale the data (highly memory-intensive).

Value

A RNA object after QC, filtering, and normalization.

Examples

```
# 1. Create mock data (include mock mitochondrial genes)
set.seed(123)
mock_counts <- matrix(rpois(100, lambda = 10), nrow = 10, ncol = 10)
rownames(mock_counts) <- c("MT-ND1", "MT-ND2", paste0("Gene_", 3:10))
colnames(mock_counts) <- paste0("Cell_", 1:10)

# 2. Build basic object
rna_obj <- Build_RNAObject(mock_counts, min_cells = 0, min_features = 0)

# 3. Run QC and Normalization (use relaxed thresholds for mock data)
rna_obj <- ProcessQC_RNA(
  obj = rna_obj,
  mt_pattern = "^MT-",
  min_nCount = 0, min_nFeature = 0, max_mt = 100,
  norm_method = "LogNormalize",
  do_scale = FALSE # Keep false to save time in example
)
dim(rna_obj$assays$RNA$data)
```

QC_epiData *Two-Step Quality Control for Epigenetic Matrix*

Description

Performs a two-step QC on a methylation level matrix. Step 1: Sorts rows (regions) by the number of NA values in ascending order based ONLY on 'level' columns, and selects the top N rows (highest quality regions). Step 2: Using ONLY the selected top rows, calculates the NA ratio for each sample's 'level' column. Retains samples with an NA ratio strictly less than the specified threshold. Finally, it keeps the 'meth', 'nonmeth', and 'level' columns for all samples that passed the QC.

Usage

```
QC_epiData(
  data,
  top_n_rows = 5000,
  max_col_na_ratio = 0.8,
  level_suffix = ".level",
  meth_suffix = ".meth",
  nonmeth_suffix = ".nonmeth"
)
```

Arguments

data	A data.frame or data.table containing the extracted metric (e.g., from extract_epi_metric). Must contain a 'region_id' column.
top_n_rows	Integer. The number of top rows (regions with the fewest NAs) to retain. Default is 5000.
max_col_na_ratio	Numeric. The maximum allowed NA ratio for columns (samples). Should be between 0 and 1. Default is 0.8 (i.e., < 80% NAs).
level_suffix	Character. The suffix identifying the 'level' columns. Default is "_level".
meth_suffix	Character. The suffix identifying the 'meth' columns. Default is "_meth".
nonmeth_suffix	Character. The suffix identifying the 'nonmeth' columns. Default is "_nonmeth".

Value

A filtered data.table retaining meth, nonmeth, and level columns for passed samples.

Examples

```
# Create mock data with NA values
mock_qc_data <- data.frame(
  region_id = c("reg1", "reg2", "reg3"),
  # Sample 1 is perfect
  S1.meth = c(10, 20, 30), S1.nonmeth = c(2, 4, 6), S1.level = c(0.8, 0.8, 0.8),
  # Sample 2 has 100% NAs (should be filtered out)
  S2.meth = c(NA, NA, NA), S2.nonmeth = c(NA, NA, NA), S2.level = c(NA, NA, NA)
)

# Run QC (Require max 50% NAs per column)
qc_result <- QC_epiData(mock_qc_data, top_n_rows = 2, max_col_na_ratio = 0.5)
head(qc_result)
```

Reduce_epiDims *Run Dimensionality Reduction on Epigenetic Level Data*

Description

Preprocesses methylation/accessibility level matrices, performs smart sample name matching with metadata, imputes missing values, and calculates dimensionality reduction coordinates (PCA, UMAP, MDS, or NMF).

Usage

```
Reduce_epiDims(
  mat,
  meta,
  sample_col,
  group_col,
  dr_method = c("PCA", "UMAP", "MDS", "NMF"),
  impute_method = c("mean", "knn"),
  knn_k = 10,
  umap_neighbors = 15,
  nmf_rank = 2
)
```

Arguments

mat	Data.frame. The level data matrix. The first column MUST be the genomic region (e.g., Chromosome regions). Rows are regions, columns are samples.
meta	Data.frame. Metadata containing sample IDs and grouping information.
sample_col	Character. Column name in ‘meta‘ representing the sample IDs.
group_col	Character. Column name in ‘meta‘ representing the sample groups/conditions.
dr_method	Character. Dimensionality reduction method. Options: "PCA", "UMAP", "MDS", "NMF". (Default: "PCA").
impute_method	Character. Method for handling NA values. Options: "knn", "mean". (Default: "mean").
knn_k	Integer. Number of neighbors to use if impute_method = "knn". (Default: 10).
umap_neighbors	Integer. Number of neighbors for UMAP. (Default: 15).
nmf_rank	Integer. Rank for NMF factorization. (Default: 2).

Value

A data.frame containing sample IDs, Dim1, Dim2, and original metadata columns, ready for plotting.

Examples

```
# 1. Create mock methylation level matrix
mock_mat <- data.frame(
  region_id = c("reg1", "reg2", "reg3", "reg4"),
  Sample1 = c(0.9, 0.8, 0.9, 0.8),
  Sample2 = c(0.8, 0.9, 0.8, 0.9),
  Sample3 = c(0.1, 0.2, 0.1, 0.2),
  Sample4 = c(0.2, 0.1, 0.2, 0.1)
)

# 2. Create mock metadata
mock_meta <- data.frame(
  ID = c("Sample1", "Sample2", "Sample3", "Sample4"),
  Group = c("Tumor", "Tumor", "Normal", "Normal")
)

# 3. Run PCA dimensionality reduction
pca_res <- Reduce_epiDims(mat = mock_mat, meta = mock_meta,
                          sample_col = "ID", group_col = "Group",
                          dr_method = "PCA")
head(pca_res)
```

Run_Diffanalysis *Core Algorithm for Epigenetic Differential Region Analysis*

Description

Performs differential methylation/epigenetic analysis between a target group and a control group (or the rest of the samples). It calculates level variances, fold changes, and utilizes Fisher's Exact Test on count data to determine statistical significance. Finally, it classifies regions as "Up-regulated", "Down-regulated", or "Not Sig" based on user-defined thresholds.

Usage

```
Run_Diffanalysis(
  raw_mat,
  meta,
  group_col,
  target_group,
  control_group = NULL,
  col_level,
  col_meth,
  col_nonmeth,
  effect_metric = "Diff",
  effect_th = 1,
  p_th = 0.05,
  verbose = TRUE
)
```

Arguments

raw_mat	A data.frame or matrix containing the raw epigenetic data. If the first column is of type character, it is automatically converted to row names.
meta	A data.frame containing metadata. This maps sample groups to their corresponding data columns.
group_col	Character. The column name in meta representing sample grouping.
target_group	Character. The name of the target (experimental) group.
control_group	Character or NULL. The name of the control group. If NULL, a "Target vs Rest" (1 vs All) analysis is performed.
col_level	Character. The column name in meta indicating the data columns for methylation 'level' (0-1 values).
col_meth	Character. The column name in meta indicating the data columns for 'meth' sites (counts).
col_nonmeth	Character. The column name in meta indicating the data columns for 'nonmeth' sites (counts).
effect_metric	Character. The metric used to determine up/down regulation. Options are "logFC", "Diff", or "Scores". Default is "Diff".
effect_th	Numeric. The absolute threshold for the selected effect_metric. Default is 1.
p_th	Numeric. The P-value threshold for significance. Default is 0.05.
verbose	Logical. Whether to print progress information to the console. Default is TRUE.

Value

A data.frame containing the comprehensive differential analysis results, sorted dynamically by P-value (ascending) and effect size (descending).

Examples

```
# 1. Create a mock raw data matrix (3 regions, 4 samples)
mock_raw <- data.frame(
  region_id = c("reg1", "reg2", "reg3"),
  # Target group
  S1.level=c(0.9, 0.5, 0.1), S1.meth=c(9, 5, 1), S1.nonmeth=c(1, 5, 9),
  S2.level=c(0.8, 0.6, 0.2), S2.meth=c(8, 6, 2), S2.nonmeth=c(2, 4, 8),
  # Control group
  C1.level=c(0.1, 0.5, 0.9), C1.meth=c(1, 5, 9), C1.nonmeth=c(9, 5, 1),
  C2.level=c(0.2, 0.4, 0.8), C2.meth=c(2, 4, 8), C2.nonmeth=c(8, 6, 2)
)

# 2. Create metadata to map columns
mock_meta <- data.frame(
  Sample = c("S1", "S2", "C1", "C2"),
  Group = c("Tumor", "Tumor", "Normal", "Normal"),
  level_col = c("S1.level", "S2.level", "C1.level", "C2.level"),
  meth_col = c("S1.meth", "S2.meth", "C1.meth", "C2.meth"),
  nonmeth_col = c("S1.nonmeth", "S2.nonmeth", "C1.nonmeth", "C2.nonmeth")
```

```
)

# 3. Run differential analysis
diff_res <- Run_Diffanalysis(
  raw_mat = mock_raw, meta = mock_meta, group_col = "Group",
  target_group = "Tumor", control_group = "Normal",
  col_level = "level_col", col_meth = "meth_col", col_nonmeth = "nonmeth_col",
  effect_metric = "Diff", effect_th = 0.5, p_th = 0.05, verbose = FALSE
)
head(diff_res)
```

RunClustering_RNA *Run Clustering for RNA Object*

Description

Performs cell/sample clustering using either a Graph-based (Leiden) approach or Hierarchical clustering.

Usage

```
RunClustering_RNA(
  obj,
  reduction = "pca",
  method = "graph",
  use_dims = 1:20,
  cluster_res = 0.8,
  k_neighbors = 20,
  cluster_k = 5
)
```

Arguments

obj	A RNA object (Requires prior dimensionality reduction).
reduction	Character. Which reduction to use: "pca" (recommended), "umap", or "tsne".
method	Character. Clustering algorithm: "graph" (KNN + Leiden) or "hierarchical".
use_dims	Integer vector. Dimensions to extract from the reduction (e.g., 1:20).
cluster_res	Numeric. Resolution parameter for Leiden algorithm (larger values = more clusters).
k_neighbors	Integer. Number of neighbors for K-NN graph construction.
cluster_k	Integer. Fixed number of clusters (k) for hierarchical clustering.

Value

A RNA object with Auto_Cluster added to its filter_meta.data.

Examples

```
# 1. Prepare data and run dimensionality reduction
set.seed(123)
mock_counts <- matrix(rpois(400, lambda = 10), nrow = 20, ncol = 20)
rownames(mock_counts) <- paste0("Gene_", 1:20)
colnames(mock_counts) <- paste0("Cell_", 1:20)
rna_obj <- Build_RNAObject(mock_counts, min_cells = 0, min_features = 0)
rna_obj <- ProcessQC_RNA(rna_obj, min_nCount = 0, min_nFeature = 0, do_scale = FALSE)
rna_obj <- RunDimReduction_RNA(rna_obj, method = "PCA", n_hvg = 15, pca_rank = 5)

# 2. Run Clustering (Hierarchical is very fast and stable for tiny data)
rna_obj <- RunClustering_RNA(
  obj = rna_obj,
  reduction = "pca",
  method = "hierarchical",
  cluster_k = 2
)
table(rna_obj$filter_meta.data$Auto_Cluster)
```

RunDEA_RNA *Run Differential Expression Analysis (Wilcoxon Rank Sum Test)*

Description

Extremely fast implementation of Wilcoxon Rank Sum Test for discovering differentially expressed genes between groups. Automatically handles 1-vs-1 or 1-vs-Rest comparisons.

Usage

```
RunDEA_RNA(
  obj,
  layer_name = "data",
  group_col = "Auto_Cluster",
  ident_1 = NULL,
  ident_2 = NULL,
  min_pct = 0.1,
  logfc_thresh = 0.25
)
```

Arguments

obj	A RNA object.
layer_name	Character. Data layer to use: "data" (recommended for log2FC) or "scale.data".
group_col	Character. Column name in filter_meta.data used for grouping cells.
ident_1	Character. The target cluster/group name (Required).
ident_2	Character. The control group name. If NULL, computes ident_1 vs all other groups (Rest).

min_pct	Numeric. Minimum fraction of cells expressing the gene in either group to retain for testing.
logfc_thresh	Numeric. Minimum absolute log2 fold change (or mean difference) required.

Value

A sorted data.frame of DEA results containing p-values, adjusted p-values, FC/Diff, and expression percentages.

Examples

```
# 1. Create mock data with simulated group differences
set.seed(123)
# Gene 1-5 will be highly expressed in Cell 1-10 (Cluster_A)
mat_A <- matrix(rpois(100, lambda = 20), nrow = 10, ncol = 10)
mat_B <- matrix(rpois(100, lambda = 1), nrow = 10, ncol = 10)
mock_counts <- rbind(cbind(mat_A, mat_B), cbind(mat_B, mat_A))
rownames(mock_counts) <- paste0("Gene_", 1:20)
colnames(mock_counts) <- paste0("Cell_", 1:20)

# 2. Build object and manually inject cluster labels for testing
rna_obj <- Build_RNAObject(mock_counts, min_cells = 0, min_features = 0)
rna_obj <- ProcessQC_RNA(rna_obj, min_nCount = 0, min_nFeature = 0, do_scale = FALSE)
rna_obj$filter_meta.data$Auto_Cluster <- rep(c("Cluster_A", "Cluster_B"), each = 10)

# 3. Run Differential Expression Analysis
dea_res <- RunDEA_RNA(
  obj = rna_obj,
  group_col = "Auto_Cluster",
  ident_1 = "Cluster_A",
  ident_2 = "Cluster_B",
  min_pct = 0, logfc_thresh = 0 # Relaxed for mock data
)
head(dea_res)
```

RunDimReduction_RNA *Run Dimensionality Reduction (PCA, t-SNE, UMAP)*

Description

Computes Highly Variable Genes (HVG) and performs PCA, t-SNE, or UMAP on the specified data layer.

Usage

```
RunDimReduction_RNA(
  obj,
  method = "PCA",
  layer_name = "data",
```

```
  n_hvg = 2000,
  pca_rank = 50,
  use_dims = 1:20,
  tsne_perp = 30,
  tsne_iter = 1000,
  umap_neighbors = 30,
  umap_mindist = 0.3
)
```

Arguments

obj	A RNA object.
method	Character. Dimensionality reduction method: "PCA", "t-SNE", or "UMAP".
layer_name	Character. Data layer to use: "data" (recommended), "scale.data", "counts", or "filter_counts".
n_hvg	Integer. Number of highly variable genes to calculate (for PCA only).
pca_rank	Integer. Number of principal components to compute.
use_dims	Integer vector. Which PCs to use for t-SNE/UMAP downstream (e.g., 1:20).
tsne_perp	Numeric. Perplexity parameter for t-SNE.
tsne_iter	Integer. Maximum iterations for t-SNE.
umap_neighbors	Integer. Number of nearest neighbors for UMAP.
umap_mindist	Numeric. Minimum distance parameter for UMAP.

Value

A RNA object with embedded coordinates.

Examples

```
# 1. Generate object and process QC
set.seed(42)
mock_counts <- matrix(rnbinom(400, mu = 10, size = 1), nrow = 20, ncol = 20)
rownames(mock_counts) <- paste0("Gene_", 1:20)
colnames(mock_counts) <- paste0("Cell_", 1:20)
rna_obj <- Build_RNAObject(mock_counts, min_cells = 0, min_features = 0)
rna_obj <- ProcessQC_RNA(rna_obj, min_nCount = 0, min_nFeature = 0, do_scale = FALSE)

# 2. Run PCA dimensionality reduction
rna_obj <- RunDimReduction_RNA(
  obj = rna_obj,
  method = "PCA",
  n_hvg = 10,
  pca_rank = 3 # Use small rank for tiny mock data
)
head(rna_obj$reductions$pca)
```

RunPseudotime_RNA *Run Pseudotime and Trajectory Inference*

Description

Computes cellular pseudotime using either a cluster-based Principal Curve method or a graph-based K-NN shortest path method.

Usage

```
RunPseudotime_RNA(
  obj,
  reduction = "umap",
  group_col = "Auto_Cluster",
  start_clus = NULL,
  algorithm = "cluster"
)
```

Arguments

obj	A RNA object.
reduction	Character. Embedding to use: "umap" (recommended), "tsne", or "pca".
group_col	Character. Column name in filter_meta.data defining clusters. Defaults to "Auto_Cluster".
start_clus	Character. Name of the root/starting cluster (Required).
algorithm	Character. Trajectory algorithm: "cluster" (Principal Curve) or "graph" (KNN Shortest Path).

Value

A RNA object with Pseudotime stored in filter_meta.data and detailed trajectory info in reductions$pseudotime.

Examples

```
# 1. Prepare fully processed mock object
set.seed(42)
mock_counts <- matrix(rpois(400, lambda = 5), nrow = 20, ncol = 20)
rownames(mock_counts) <- paste0("Gene_", 1:20)
colnames(mock_counts) <- paste0("Cell_", 1:20)
rna_obj <- Build_RNAObject(mock_counts, min_cells = 0, min_features = 0)
rna_obj <- ProcessQC_RNA(rna_obj, min_nCount = 0, min_nFeature = 0, do_scale = FALSE)
rna_obj <- RunDimReduction_RNA(rna_obj, method = "PCA", n_hvg = 15, pca_rank = 3)
rna_obj <- RunClustering_RNA(rna_obj, reduction = "pca", method = "hierarchical", cluster_k = 2)

# 2. Run Pseudotime Inference using PCA space
rna_obj <- RunPseudotime_RNA(
  obj = rna_obj,
```

```
  reduction = "pca",
  group_col = "Auto_Cluster",
  start_clus = "Cluster_1",
  algorithm = "cluster"
)
head(rna_obj$filter_meta.data$Pseudotime)
```

Index

AddMetaData_RNA, 2
Aggregate_epiRegions, 3

Build_RNAObject, 4

CreateRNAObject, 6

Extract_epiData, 7

Integrate_MultiOmics, 8

PlotDimRed_Epi, 10
PlotDimRed_RNA, 11
PlotLandscape_Epi, 12
PlotOmicsMatrix, 13
PlotOmicsScatter, 14
PlotPseudo_RNA, 16
PlotQC_RNA, 17
PlotVolcano_Epi, 18
PlotVolcano_RNA, 19
ProcessQC_RNA, 20

QC_epiData, 21

Reduce_epiDims, 23
Run_Diffanalysis, 24
RunClustering_RNA, 26
RunDEA_RNA, 27
RunDimReduction_RNA, 28
RunPseudotime_RNA, 30