Package 'blase'

November 11, 2026

Title Bulk Linking Analysis for Single-cell Experiments

Version 1.2.1

Description BLASE is a method for finding where bulk RNA-seq data lies on a single-cell pseudotime trajectory. It uses a fast and understandable approach based on Spearman correlation, with bootstrapping to provide confidence. BLASE can be used to ``date" bulk RNA-seq data, annotate cell types in scRNA-seq, and help correct for developmental phenotype differences in bulk RNA-seq experiments.

License GPL (>= 3)

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

biocViews Transcriptomics, SingleCell, Sequencing, GeneExpression, Transcription, RNASeq, TimeCourse, CellBiology, Software, CellBasedAssays

Imports SummarizedExperiment, SingleCellExperiment, ggplot2, viridis, patchwork, Matrix, scater, methods, rlang, BiocParallel, boot, dplyr, mgcv, stats, MatrixGenerics, Seurat (>= 4.0.0), lsa

Suggests knitr, rmarkdown, testthat (>= 3.2.3), covr, tradeSeq, scran, slingshot, tools, reshape2, plyr, fs, sparseMatrixStats, ggVennDiagram, uwot, BiocStyle, DelayedMatrixStats, limma

Config/testthat/edition 3

VignetteBuilder knitr

URL https://andrewmccluskey-uog.github.io/blase/

BugReports https://andrewmccluskey-uog.github.io/blase/issues

Depends R (>= 4.5.0)

LazyData false

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

git_branch RELEASE_3_23

git_last_commit 63e0c7a

git_last_commit_date 2026-08-11

Repository Bioconductor 3.23

Date/Publication 2026-11-11

Author Andrew McCluskey [aut, cre] (ORCID:
<https://orcid.org/0009-0004-4187-799X>),
Toby Kettlewell [aut] (ORCID: <https://orcid.org/0009-0001-1225-3318>),
Adrian M. Smith [aut] (ORCID: <https://orcid.org/0000-0001-8833-2330>),
Rhiannon Kundu [aut] (ORCID: <https://orcid.org/0000-0003-3970-5860>),
David A. Gunn [aut] (ORCID: <https://orcid.org/0000-0001-9866-3221>),
Thomas D. Otto [aut, ths] (ORCID:
<https://orcid.org/0000-0002-1246-7404>)

Maintainer Andrew McCluskey <2117532m@student.gla.ac.uk>

Contents

annotate_sce . . . 3
as.BlaseData . . . 4
assign_pseudotime_bins . . . 5
best_bin . . . 8
best_correlation . . . 9
bins . . . 11
bins<- . . . 11
BlaseData-class . . . 12
bootstrap_iterations . . . 13
bulk_name . . . 14
bulk_name<- . . . 16
calculate_gene_peakedness . . . 16
evaluate_parameters . . . 18
evaluate_top_n_genes . . . 20
find_best_params . . . 21
genes . . . 23
genes<- . . . 23
gene_peakedness_spread_selection . . . 24
get_bins_as_bulk . . . 26
get_top_n_genes . . . 27
MappingResult . . . 28
mapping_history . . . 30
map_all_best_bins . . . 31
map_best_bin . . . 33
MCA_PF_SCE . . . 35
metric . . . 36
painter_microarray . . . 37
plot_bin_population . . . 38
plot_find_best_params_results . . . 39
plot_gene_peakedness . . . 41
plot_mapping_result . . . 42

plot_mapping_result_corr . . . 44
plot_mapping_result_heatmap . . . 45
PRIVATE_.ci . . . 47
PRIVATE_correlation.ci . . . 48
PRIVATE_cosine_similarity.ci . . . 49
PRIVATE_distance.ci . . . 49
pseudobulk_bins . . . 50
pseudobulk_bins<- . . . 51
show,BlaseData-method . . . 51
show,MappingResult-method . . . 52
smooth_gene . . . 53
strong_mapping . . . 55
top_2_distance . . . 57
tradeSeq_BLASE_example_sce . . . 58
zhang_2021_heat_shock_bulk . . . 59

Index **60**

`annotate_sce` *Annotate a SCE with BLASE Mappings*

Description

Annotates an SCE with the names of bulk samples that best match each pseudotime bin. For each pseudotime bin, we find the highest correlation with a bulk sample that was mapped against it. Because of this approach, a bulk which mapped best to another pseudotime bin may be the best correlation with the current pseudotime bin of interest.

Usage

```
annotate_sce(
  sce,
  blase_results,
  annotation_col = "BLASE_Annotation",
  include_stats = FALSE
)
```

Arguments

`sce`	The SingleCellExperiment::SingleCellExperiment to annotate.
`blase_results`	A list of MappingResult to use for the annotation.
`annotation_col`	String. The name of the metadata column in which to store the new annotations.
`include_stats`	Boolean. Whether or not to include metadata columns containing the correlation of the best matching bin, and whether that mapping was strong.

Value

A SingleCellExperiment::SingleCellExperiment with annotations added to metadata (in a column defined by `annotation_col`), and the correlations in `BLASE_Annotation_Correlation` if `include_stats` is enabled.

Examples

```
counts_matrix <- matrix(
    c(seq_len(120) / 10, seq_len(120) / 5),
    ncol = 48, nrow = 5
)
sce <- SingleCellExperiment::SingleCellExperiment(assays = list(
    normcounts = counts_matrix, logcounts = log(counts_matrix)
))
colnames(sce) <- seq_len(48)
rownames(sce) <- as.character(seq_len(5))
sce$cell_type <- c(rep("celltype_1", 24), rep("celltype_2", 24))

sce$pseudotime <- seq_len(48) - 1
blase_data <- as.BlaseData(sce, pseudotime_slot = "pseudotime", n_bins = 4)
genes(blase_data) <- as.character(seq_len(5))

bulk_counts <- matrix(seq_len(15) * 10, ncol = 3, nrow = 5)
colnames(bulk_counts) <- c("A", "B", "C")
rownames(bulk_counts) <- as.character(seq_len(5))

# Map all bulks to bin
results <- map_all_best_bins(blase_data, bulk_counts)

sce <- assign_pseudotime_bins(
    sce,
    pseudotime_slot = "pseudotime", n_bins = 4
)

# Annotate SC from existing bulk
sce <- annotate_sce(sce, results)
table(sce$BLASE_Annotation)
```

`as.BlaseData` *Conversion to BlaseData*

Description

Conversion to BlaseData

Usage

```
as.BlaseData(x, ...)
```

```
## S4 method for signature 'SingleCellExperiment'
as.BlaseData(
  x,
  pseudotime_slot = "slingPseudotime_1",
  n_bins = 20,
  split_by = "pseudotime_range"
)
```

Arguments

x	An object to take counts from
...	additional arguments passed to object-specific methods.
pseudotime_slot	String or vector of strings. The SingleCellExperiment::SingleCellExperiment slot(s) containing pseudotime values for each cell to be passed to `assign_pseudotime_bins()`.
n_bins	Integer. The number of bins to create, passed to `assign_pseudotime_bins()`.
split_by	String. The split_by method to be passed on to `assign_pseudotime_bins()`. Must be one of `pseudotime_range` or `cells`.

Value

An BlaseData object

Examples

```
counts <- matrix(rpois(100, lambda = 10), ncol = 10, nrow = 10)
sce <- SingleCellExperiment::SingleCellExperiment(
    assays = list(normcounts = counts)
)
sce$pseudotime <- seq_len(10) - 1
data <- as.BlaseData(sce, pseudotime_slot = "pseudotime", n_bins = 3)
genes(data) <- as.character(seq_len(10))

genes(data)
```

---

assign_pseudotime_bins

Assign Pseudotime Bins to a source object's metadata

Description

Assign Pseudotime Bins to a source object's metadata

Usage

```
assign_pseudotime_bins(
  x,
  split_by = "pseudotime_range",
  n_bins = 20,
  pseudotime_slot = "slingPseudotime_1",
  ...
)

## S4 method for signature 'SingleCellExperiment'
assign_pseudotime_bins(
  x,
  split_by,
  n_bins,
  pseudotime_slot = "slingPseudotime_1"
)

## S4 method for signature 'data.frame'
assign_pseudotime_bins(
  x,
  split_by,
  n_bins,
  pseudotime_slot = "slingPseudotime_1"
)

## S4 method for signature 'Seurat'
assign_pseudotime_bins(
  x,
  split_by,
  n_bins,
  pseudotime_slot = "slingPseudotime_1"
)
```

Arguments

x	An object to add metadata to.
split_by	String. The technique used to split the bins. The default `pseudotime_range` picks the bin for a cell based on a constant range of pseudotime. `cells` picks the bin for a cell based on an even number of cells per bin.
n_bins	Integer. The number of bins to split the cells into.
pseudotime_slot	String or Vector of Strings. The name of the SingleCellExperiment::SingleCellExperiment slot(s) containing the pseudotime values for each cell.
...	For arguments passed to other functions. Unused.

Value

A copy of x where cells are annotated with their pseudotime bin.

Examples

```
counts_matrix <- matrix(
    c(seq_len(120) / 10, seq_len(120) / 5),
    ncol = 48, nrow = 5
)
sce <- SingleCellExperiment::SingleCellExperiment(assays = list(
    normcounts = counts_matrix, logcounts = log(counts_matrix)
))
colnames(sce) <- seq_len(48)
rownames(sce) <- as.character(seq_len(5))
sce$cell_type <- c(rep("celltype_1", 24), rep("celltype_2", 24))

sce$pseudotime <- seq_len(48) - 1
blase_data <- as.BlaseData(sce, pseudotime_slot = "pseudotime", n_bins = 4)
genes(blase_data) <- as.character(seq_len(5))

bulk_counts <- matrix(seq_len(15) * 10, ncol = 3, nrow = 5)
colnames(bulk_counts) <- c("A", "B", "C")
rownames(bulk_counts) <- as.character(seq_len(5))

# Map to bin
result <- map_best_bin(blase_data, "B", bulk_counts)
result

# Map all bulks to bin
results <- map_all_best_bins(blase_data, bulk_counts)

# Plot Heatmap
plot_mapping_result_heatmap(list(result))

# Plot Correlation
plot_mapping_result_corr(result)

# Plot populations
sce <- assign_pseudotime_bins(
    sce,
    pseudotime_slot = "pseudotime", n_bins = 4
)
plot_bin_population(sce, best_bin(result), group_by_slot = "cell_type")

# Getters
bulk_name(result)
best_bin(result)
best_correlation(result)
top_2_distance(result)
strong_mapping(result)
mapping_history(result)
bootstrap_iterations(result)
metric(result)

# Setters
bulk_name(result) <- "New Name"
```

best_bin *Get best bin of a BLASE Mapping Results object.*

Description

Get best bin of a BLASE Mapping Results object.

Usage

```
best_bin(x)

## S4 method for signature 'MappingResult'
best_bin(x)
```

Arguments

x	a MappingResult object

Value

Integer. The best bin ID of this mapping

Examples

```
counts_matrix <- matrix(
    c(seq_len(120) / 10, seq_len(120) / 5),
    ncol = 48, nrow = 5
)
sce <- SingleCellExperiment::SingleCellExperiment(assays = list(
    normcounts = counts_matrix, logcounts = log(counts_matrix)
))
colnames(sce) <- seq_len(48)
rownames(sce) <- as.character(seq_len(5))
sce$cell_type <- c(rep("celltype_1", 24), rep("celltype_2", 24))

sce$pseudotime <- seq_len(48) - 1
blase_data <- as.BlaseData(sce, pseudotime_slot = "pseudotime", n_bins = 4)
genes(blase_data) <- as.character(seq_len(5))

bulk_counts <- matrix(seq_len(15) * 10, ncol = 3, nrow = 5)
colnames(bulk_counts) <- c("A", "B", "C")
rownames(bulk_counts) <- as.character(seq_len(5))

# Map to bin
result <- map_best_bin(blase_data, "B", bulk_counts)
result

# Map all bulks to bin
results <- map_all_best_bins(blase_data, bulk_counts)
```

```
# Plot Heatmap
plot_mapping_result_heatmap(list(result))

# Plot Correlation
plot_mapping_result_corr(result)

# Plot populations
sce <- assign_pseudotime_bins(
    sce,
    pseudotime_slot = "pseudotime", n_bins = 4
)
plot_bin_population(sce, best_bin(result), group_by_slot = "cell_type")

# Getters
bulk_name(result)
best_bin(result)
best_correlation(result)
top_2_distance(result)
strong_mapping(result)
mapping_history(result)
bootstrap_iterations(result)
metric(result)

# Setters
bulk_name(result) <- "New Name"
```

`best_correlation` *Get best correlation of a BLASE Mapping Results object.*

Description

Get best correlation of a BLASE Mapping Results object.

Usage

```
best_correlation(x)

## S4 method for signature 'MappingResult'
best_correlation(x)
```

Arguments

x	a MappingResult object

Value

Decimal. The highest correlation value of this mapping

Examples

```
counts_matrix <- matrix(
    c(seq_len(120) / 10, seq_len(120) / 5),
    ncol = 48, nrow = 5
)
sce <- SingleCellExperiment::SingleCellExperiment(assays = list(
    normcounts = counts_matrix, logcounts = log(counts_matrix)
))
colnames(sce) <- seq_len(48)
rownames(sce) <- as.character(seq_len(5))
sce$cell_type <- c(rep("celltype_1", 24), rep("celltype_2", 24))

sce$pseudotime <- seq_len(48) - 1
blase_data <- as.BlaseData(sce, pseudotime_slot = "pseudotime", n_bins = 4)
genes(blase_data) <- as.character(seq_len(5))

bulk_counts <- matrix(seq_len(15) * 10, ncol = 3, nrow = 5)
colnames(bulk_counts) <- c("A", "B", "C")
rownames(bulk_counts) <- as.character(seq_len(5))

# Map to bin
result <- map_best_bin(blase_data, "B", bulk_counts)
result

# Map all bulks to bin
results <- map_all_best_bins(blase_data, bulk_counts)

# Plot Heatmap
plot_mapping_result_heatmap(list(result))

# Plot Correlation
plot_mapping_result_corr(result)

# Plot populations
sce <- assign_pseudotime_bins(
    sce,
    pseudotime_slot = "pseudotime", n_bins = 4
)
plot_bin_population(sce, best_bin(result), group_by_slot = "cell_type")

# Getters
bulk_name(result)
best_bin(result)
best_correlation(result)
top_2_distance(result)
strong_mapping(result)
mapping_history(result)
bootstrap_iterations(result)
metric(result)

# Setters
bulk_name(result) <- "New Name"
```

bins *Get bins of a BLASE Data object.*

Description

Get bins of a BLASE Data object.

Usage

```
bins(x)

## S4 method for signature 'BlaseData'
bins(x)
```

Arguments

x	a BlaseData object

Value

vector of integers. The bin names in the BLASE Data object.

Examples

```
counts <- matrix(rpois(100, lambda = 10), ncol = 10, nrow = 10)
sce <- SingleCellExperiment::SingleCellExperiment(
    assays = list(normcounts = counts)
)
sce$pseudotime <- seq_len(10) - 1
data <- as.BlaseData(sce, pseudotime_slot = "pseudotime", n_bins = 3)
genes(data) <- as.character(seq_len(10))

genes(data)
```

bins<- *Set genes of a BLASE Data object.*

Description

Set genes of a BLASE Data object.

Usage

```
bins(x) <- value

## S4 replacement method for signature 'BlaseData'
bins(x) <- value
```

Arguments

x	a BlaseData object
value	Vector of integers The new value for bins slot

Value

Nothing

BlaseData-class *Blase Data Object*

Description

For creation details, see as.BlaseData()

Value

A BlaseData object

Slots

pseudobulk_bins list of data.frames. Each item is a normalised count matrix representing a bin, where a column is a cell in the bin and each row is a gene.

bins list. A list of bin names for each timepoint.

genes list. A list of the genes selected for discriminating timepoints.

Examples

```
counts <- matrix(rpois(100, lambda = 10), ncol = 10, nrow = 10)
sce <- SingleCellExperiment::SingleCellExperiment(
    assays = list(normcounts = counts)
)
sce$pseudotime <- seq_len(10) - 1
data <- as.BlaseData(sce, pseudotime_slot = "pseudotime", n_bins = 3)
genes(data) <- as.character(seq_len(10))

genes(data)
```

`bootstrap_iterations` *Get the number of bootstrap iterations performed for a BLASE Mapping Results object.*

Description

Get the number of bootstrap iterations performed for a BLASE Mapping Results object.

Usage

```
bootstrap_iterations(x)

## S4 method for signature 'MappingResult'
bootstrap_iterations(x)
```

Arguments

x	a MappingResult object

Value

Integer. The number of iterations performed for this mapping.

Examples

```
counts_matrix <- matrix(
    c(seq_len(120) / 10, seq_len(120) / 5),
    ncol = 48, nrow = 5
)
sce <- SingleCellExperiment::SingleCellExperiment(assays = list(
    normcounts = counts_matrix, logcounts = log(counts_matrix)
))
colnames(sce) <- seq_len(48)
rownames(sce) <- as.character(seq_len(5))
sce$cell_type <- c(rep("celltype_1", 24), rep("celltype_2", 24))

sce$pseudotime <- seq_len(48) - 1
blase_data <- as.BlaseData(sce, pseudotime_slot = "pseudotime", n_bins = 4)
genes(blase_data) <- as.character(seq_len(5))

bulk_counts <- matrix(seq_len(15) * 10, ncol = 3, nrow = 5)
colnames(bulk_counts) <- c("A", "B", "C")
rownames(bulk_counts) <- as.character(seq_len(5))

# Map to bin
result <- map_best_bin(blase_data, "B", bulk_counts)
result

# Map all bulks to bin
```

```
results <- map_all_best_bins(blase_data, bulk_counts)

# Plot Heatmap
plot_mapping_result_heatmap(list(result))

# Plot Correlation
plot_mapping_result_corr(result)

# Plot populations
sce <- assign_pseudotime_bins(
    sce,
    pseudotime_slot = "pseudotime", n_bins = 4
)
plot_bin_population(sce, best_bin(result), group_by_slot = "cell_type")

# Getters
bulk_name(result)
best_bin(result)
best_correlation(result)
top_2_distance(result)
strong_mapping(result)
mapping_history(result)
bootstrap_iterations(result)
metric(result)

# Setters
bulk_name(result) <- "New Name"
```

---

bulk_name *Get name of bulk of a BLASE Mapping Results object.*

Description

Get name of bulk of a BLASE Mapping Results object.

Usage

```
bulk_name(x)

## S4 method for signature 'MappingResult'
bulk_name(x)
```

Arguments

x	a MappingResult object

Value

String. The name of the bulk used to map against.

Examples

```
counts_matrix <- matrix(
    c(seq_len(120) / 10, seq_len(120) / 5),
    ncol = 48, nrow = 5
)
sce <- SingleCellExperiment::SingleCellExperiment(assays = list(
    normcounts = counts_matrix, logcounts = log(counts_matrix)
))
colnames(sce) <- seq_len(48)
rownames(sce) <- as.character(seq_len(5))
sce$cell_type <- c(rep("celltype_1", 24), rep("celltype_2", 24))

sce$pseudotime <- seq_len(48) - 1
blase_data <- as.BlaseData(sce, pseudotime_slot = "pseudotime", n_bins = 4)
genes(blase_data) <- as.character(seq_len(5))

bulk_counts <- matrix(seq_len(15) * 10, ncol = 3, nrow = 5)
colnames(bulk_counts) <- c("A", "B", "C")
rownames(bulk_counts) <- as.character(seq_len(5))

# Map to bin
result <- map_best_bin(blase_data, "B", bulk_counts)
result

# Map all bulks to bin
results <- map_all_best_bins(blase_data, bulk_counts)

# Plot Heatmap
plot_mapping_result_heatmap(list(result))

# Plot Correlation
plot_mapping_result_corr(result)

# Plot populations
sce <- assign_pseudotime_bins(
    sce,
    pseudotime_slot = "pseudotime", n_bins = 4
)
plot_bin_population(sce, best_bin(result), group_by_slot = "cell_type")

# Getters
bulk_name(result)
best_bin(result)
best_correlation(result)
top_2_distance(result)
strong_mapping(result)
mapping_history(result)
bootstrap_iterations(result)
metric(result)

# Setters
bulk_name(result) <- "New Name"
```

bulk_name<- *Set name of bulk of a BLASE Mapping Results object.*

Description

Set name of bulk of a BLASE Mapping Results object.

Usage

```
bulk_name(x) <- value

## S4 replacement method for signature 'MappingResult'
bulk_name(x) <- value
```

Arguments

x	a MappingResult object
value	String. The name of the bulk used to map against.

Value

Nothing

Examples

```
counts <- matrix(rpois(100, lambda = 10), ncol = 10, nrow = 10)
sce <- SingleCellExperiment::SingleCellExperiment(
    assays = list(normcounts = counts)
)
sce$pseudotime <- seq_len(10) - 1
data <- as.BlaseData(sce, pseudotime_slot = "pseudotime", n_bins = 3)
genes(data) <- as.character(seq_len(10))

genes(data)
```

calculate_gene_peakedness *calculate_gene_peakedness*

Description

Calculate the peakedness of a gene. The power is the ratio of the mean of reads 5% either side of the smoothed peak of the gene's expression over pseudotime against the mean of the reads outside of this.

This function can take some time to complete, please be patient.

Usage

```
calculate_gene_peakedness(
  sce,
  window_pct = 10,
  pseudotime_slot = "slingPseudotime_1",
  knots = 10,
  BPPARAM = BiocParallel::SerialParam()
)
```

Arguments

sce	SingleCellExperiment::SingleCellExperiment to do the calculations on.
window_pct	Decimal between 0-100. The size of the window to consider, as a percentage of the maximum pseudotime value.
pseudotime_slot	String. The name of the metadata column in the SCE object containing pseudotime
knots	Integer. The number of knots to use when fitting the GAM
BPPARAM	The BiocParallel::BiocParallelParam for parallelisation. Defaults to BiocParallel::SerialParam.

Value

Dataframe, where each row is a gene, and the following columns: mean_expression_in_window (decimal), mean_expression_out_window (decimal), ratio (decimal)

Examples

```
ncells <- 70
ngenes <- 100
# Each gene should have mean around its gene number
counts <- c()
for (i in seq_len(ngenes)) {
    counts <- c(counts, dnorm(seq_len(ncells), mean = (ncells / i), sd = 1))
}

counts_matrix <- matrix(
    counts,
    ncol = ncells,
    nrow = ngenes
)
sce <- SingleCellExperiment::SingleCellExperiment(assays = list(
    counts = counts_matrix * 3,
    normcounts = counts_matrix,
    logcounts = log(counts_matrix)
))
colnames(sce) <- paste0("cell", seq_len(ncells))
rownames(sce) <- paste0("gene", seq_len(ngenes))
sce$cell_type <- c(
```

```
    rep("celltype_1", ncells / 2),
    rep("celltype_2", ncells / 2)
)

sce$pseudotime <- seq_len(ncells) - 1
genelist <- rownames(sce)

# calculate_gene_peakedness
gene_peakedness <- calculate_gene_peakedness(
    sce,
    pseudotime_slot = "pseudotime"
)

head(gene_peakedness)

# plot_gene_peakedness
plot_gene_peakedness(sce, gene_peakedness, "gene20",
    pseudotime_slot = "pseudotime"
)

# smooth_gene
smoothed_gene20 <- smooth_gene(
    sce, "gene20",
    pseudotime_slot = "pseudotime"
)
head(smoothed_gene20)

# Select best spread of genes
genes_to_use <- gene_peakedness_spread_selection(sce, gene_peakedness,
    genes_per_bin = 2, n_gene_bins = 1, pseudotime_slot = "pseudotime"
)

print(genes_to_use)
plot(
    x = gene_peakedness[
        gene_peakedness$gene %in% genes_to_use, "peak_pseudotime"
    ],
    y = gene_peakedness[gene_peakedness$gene %in% genes_to_use, "ratio"]
)
```

`evaluate_parameters` *Evaluate n_bins and n_genes for bin mapping*

Description

Will use the n_bins and n_genes implied by the `sce` and `pseudotime_bins_top_n_genes_df` parameters and return quality metrics and an optional chart.

Usage

```
evaluate_parameters(
  blase_data,
  bootstrap_iterations = 200,
  BPPARAM = BiocParallel::SerialParam(),
  make_plot = FALSE,
  plot_columns = 4
)
```

Arguments

blase_data	The BlaseData object to use.
bootstrap_iterations	Integer. Iterations for bootstrapping when calculating strong mappings.
BPPARAM	The BiocParallel::BiocParallelParam configuration. Defaults to BiocParallel::SerialParam
make_plot	Boolean. Whether or not to render the plot showing the correlations for each pseudobulk bin when we try to map the given bin.
plot_columns	Integer. How many columns to use in the plot.

Value

A vector of length 3:

- "worst top 2 distance" decimal containing the lowest difference between the absolute values of the top 2 most correlated bins for each bin. Higher is better for differentiating.
- "mean top 2 distance" decimal containing the mean top 2 distance across the entire set of genes and bins. Higher is better for differentiation, but it should matter less than the worst value.
- "strong_mapping_pct" decimal from 0-1. The percent of mappings for this setup which were annotated as strong by BLASE.

Examples

```
ncells <- 70
ngenes <- 100
counts_matrix <- matrix(
    c(seq_len(3500) / 10, seq_len(3500) / 5),
    ncol = ncells,
    nrow = ngenes
)
sce <- SingleCellExperiment::SingleCellExperiment(assays = list(
    normcounts = counts_matrix, logcounts = log(counts_matrix)
))
colnames(sce) <- seq_len(ncells)
rownames(sce) <- as.character(seq_len(ngenes))
sce$cell_type <- c(
    rep("celltype_1", ncells / 2),
    rep("celltype_2", ncells / 2)
)
```

```
sce$pseudotime <- seq_len(ncells) - 1
genelist <- as.character(seq_len(ngenes))

# Evaluating created BlaseData
blase_data <- as.BlaseData(sce, pseudotime_slot = "pseudotime", n_bins = 10)
genes(blase_data) <- genelist[1:20]

# Check convexity of parameters
evaluate_parameters(blase_data, make_plot = TRUE)
```

`evaluate_top_n_genes` *Evaluate Top Genes*

Description

Shows plots over bins of expression of the top n genes. This is designed to help identify if you have selected genes that vary over the pseudotime you have chosen bins to exist over. Uses the normcounts of the SCE.

Usage

```
evaluate_top_n_genes(blase_data, n_genes_to_plot = 16, plot_columns = 4)
```

Arguments

`blase_data`	The BlaseData to get bins and expression from.
`n_genes_to_plot`	Integer. The number of genes to plot.
`plot_columns`	Integer. The number of columns to plot the grid with. Best as a divisor of `n_genes_to_plot`.

Value

A ggplot2::ggplot2 plot showing the normalised expression of the top genes over pseudotime bins.

Examples

```
ncells <- 70
ngenes <- 100
counts_matrix <- matrix(
    c(seq_len(3500) / 10, seq_len(3500) / 5),
    ncol = ncells,
    nrow = ngenes
)
sce <- SingleCellExperiment::SingleCellExperiment(assays = list(
    normcounts = counts_matrix, logcounts = log(counts_matrix)
))
colnames(sce) <- paste0("cell", seq_len(ncells))
rownames(sce) <- paste0("gene", seq_len(ngenes))
```

```
sce$cell_type <- c(
    rep("celltype_1", ncells / 2),
    rep("celltype_2", ncells / 2)
)

sce$pseudotime <- seq_len(ncells) - 1
genelist <- rownames(sce)

# Evaluating created BlaseData
blase_data <- as.BlaseData(sce, pseudotime_slot = "pseudotime", n_bins = 10)
genes(blase_data) <- genelist[1:20]

# Check gene expression over pseudotime
evaluate_top_n_genes(blase_data)
```

`find_best_params` *Identify the Best Parameters For Your Dataset*

Description

Identify the Best Parameters For Your Dataset

Usage

```
find_best_params(
  x,
  genelist,
  bins_count_range = c(5, 10, 20, 40),
  gene_count_range = c(10, 20, 40, 80),
  bootstrap_iterations = 200,
  BPPARAM = BiocParallel::SerialParam(),
  ...
)
```

Arguments

`x`	The object to create 'BlaseData" from
`genelist`	Vector of strings. The list of genes to use (ordered by descending goodness)
`bins_count_range`	Integer vector. The n_bins list to try out
`gene_count_range`	Integer vector. The n_genes list to try out
`bootstrap_iterations`	Integer. Iterations for bootstrapping when calculating strong mappings.
`BPPARAM`	The BiocParallel::BiocParallelParam. Defaults to BiocParallel::SerialParam
`...`	params to be passed to child functions, see `as.BlaseData()`

Value

A dataframe of the results.

- bin_count: Integer. The bin count for this attempt
- gene_count: Integer. The top n genes to use for this attempt
- min_convexity: Decimal. The worst convexity for these parameters
- mean_convexity: Decimal. The mean convexity for these parameters
- strong_mapping_pct: Decimal. The percent of bins which were strongly mapped to themselves for these parameters. If this value is low, then it is likely that in real use, few or no results will be strongly mapped.

See Also

plot_find_best_params_results() for plotting the results of this function.

Examples

```
ncells <- 70
ngenes <- 100
counts_matrix <- matrix(
    c(seq_len(3500) / 10, seq_len(3500) / 5),
    ncol = ncells,
    nrow = ngenes
)
sce <- SingleCellExperiment::SingleCellExperiment(assays = list(
    normcounts = counts_matrix, logcounts = log(counts_matrix)
))
colnames(sce) <- paste0("cell", seq_len(ncells))
rownames(sce) <- paste0("gene", seq_len(ngenes))
sce$cell_type <- c(
    rep("celltype_1", ncells / 2),
    rep("celltype_2", ncells / 2)
)

sce$pseudotime <- seq_len(ncells) - 1
genelist <- rownames(sce)

# Finding the best params for the BlaseData
best_params <- find_best_params(
    sce, genelist,
    bins_count_range = c(2, 3),
    gene_count_range = c(20, 50),
    pseudotime_slot = "pseudotime",
    split_by = "pseudotime_range"
)
best_params
plot_find_best_params_results(best_params)
```

genes *Get genes of a BLASE Data object.*

Description

Get genes of a BLASE Data object.

Usage

```
genes(x)

## S4 method for signature 'BlaseData'
genes(x)
```

Arguments

x	a BlaseData object

Value

The vector of genes a BLASE object will use for mappings.

Examples

```
counts <- matrix(rpois(100, lambda = 10), ncol = 10, nrow = 10)
sce <- SingleCellExperiment::SingleCellExperiment(
    assays = list(normcounts = counts)
)
sce$pseudotime <- seq_len(10) - 1
data <- as.BlaseData(sce, pseudotime_slot = "pseudotime", n_bins = 3)
genes(data) <- as.character(seq_len(10))

genes(data)
```

genes<- *Set genes of a BLASE Data object.*

Description

Set genes of a BLASE Data object.

Usage

```
genes(x) <- value

## S4 replacement method for signature 'BlaseData'
genes(x) <- value
```

Arguments

x	a BlaseData object
value	Vector of strings. The new value for genes slot

Value

Nothing

Examples

```
counts <- matrix(rpois(100, lambda = 10), ncol = 10, nrow = 10)
sce <- SingleCellExperiment::SingleCellExperiment(
    assays = list(normcounts = counts)
)
sce$pseudotime <- seq_len(10) - 1
data <- as.BlaseData(sce, pseudotime_slot = "pseudotime", n_bins = 3)
genes(data) <- as.character(seq_len(10))

genes(data)
```

---

gene_peakedness_spread_selection

Gene Peakedness Spread Selection

Description

This function selects genes with peaks evenly distributed from a pseudotime trajectory. It does this by splitting pseudotime into evenly spread regions of pseudotime, and then selecting genes with the highest peakedness ratio with a peak inside that region of pseudotime. The number of regions and genes per region can be tuned.

Usage

```
gene_peakedness_spread_selection(
  sce,
  gene_peakedness_df,
  genes_per_bin = 10,
  n_gene_bins = 10,
  pseudotime_slot = "slingPseudotime_1"
)
```

Arguments

sce	SingleCellExperiment::SingleCellExperiment to obtain pseudotime values from
gene_peakedness_df	Gene peakedness DF generated by `calculate_gene_peakedness()`
genes_per_bin	Integer. Number of genes to select per gene bin.

n_gene_bins Integer. Number of gene bins to create over pseudotime. We recommend around 1-2x the number of pseudotime bins you want to use.

pseudotime_slot String. The name of the pseudotime column in the SCE metadata.

Value

A list of gene IDs with the highest ratios across regions of pseudotime.

Examples

```
ncells <- 70
ngenes <- 100
# Each gene should have mean around its gene number
counts <- c()
for (i in seq_len(ngenes)) {
    counts <- c(counts, dnorm(seq_len(ncells), mean = (ncells / i), sd = 1))
}

counts_matrix <- matrix(
    counts,
    ncol = ncells,
    nrow = ngenes
)
sce <- SingleCellExperiment::SingleCellExperiment(assays = list(
    counts = counts_matrix * 3,
    normcounts = counts_matrix,
    logcounts = log(counts_matrix)
))
colnames(sce) <- paste0("cell", seq_len(ncells))
rownames(sce) <- paste0("gene", seq_len(ngenes))
sce$cell_type <- c(
    rep("celltype_1", ncells / 2),
    rep("celltype_2", ncells / 2)
)

sce$pseudotime <- seq_len(ncells) - 1
genelist <- rownames(sce)

# calculate_gene_peakedness
gene_peakedness <- calculate_gene_peakedness(
    sce,
    pseudotime_slot = "pseudotime"
)

head(gene_peakedness)

# plot_gene_peakedness
plot_gene_peakedness(sce, gene_peakedness, "gene20",
    pseudotime_slot = "pseudotime"
)
```

```
# smooth_gene
smoothed_gene20 <- smooth_gene(
    sce, "gene20",
    pseudotime_slot = "pseudotime"
)
head(smoothed_gene20)

# Select best spread of genes
genes_to_use <- gene_peakedness_spread_selection(sce, gene_peakedness,
    genes_per_bin = 2, n_gene_bins = 1, pseudotime_slot = "pseudotime"
)

print(genes_to_use)
plot(
    x = gene_peakedness[
        gene_peakedness$gene %in% genes_to_use, "peak_pseudotime"
    ],
    y = gene_peakedness[gene_peakedness$gene %in% genes_to_use, "ratio"]
)
```

`get_bins_as_bulk` *Get a pseudobulk of bins with at least 2 replicates*

Description

This function will try to create a pseudobulked count matrix for the bins. When a replicate has too few cells, it is discounted. If only one exists, then we sample from it twice to create the pseudobulks.

Usage

```
get_bins_as_bulk(
  pseudotime_sce,
  min_cells_for_bulk = 50,
  replicate_slot = "replicate"
)
```

Arguments

`pseudotime_sce`	The SingleCellExperiment::SingleCellExperiment object to get the bins from
`min_cells_for_bulk`	Integer. The minimum cells to look for per replicate and bin.
`replicate_slot`	String. The name of the matadata column in the Single Cell Experiment that contains replicate information

Value

A dataframe containing the pseudobulked counts matrix.

Examples

```
library(SingleCellExperiment, quietly = TRUE)
library(blase)
counts <- matrix(rpois(1000, lambda = 10), ncol = 100, nrow = 10)
sce <- SingleCellExperiment::SingleCellExperiment(
    assays = list(normcounts = counts, counts = counts / 2)
)
sce$pseudotime <- seq_len(100) - 1
colnames(sce) <- seq_len(100)
rownames(sce) <- as.character(seq_len(10))
sce <- assign_pseudotime_bins(sce,
    n_bins = 5,
    pseudotime_slot = "pseudotime", split_by = "cells"
)
sce$replicate <- rep(c(1, 2), 50)
result <- get_bins_as_bulk(
    sce,
    min_cells_for_bulk = 1,
    replicate_slot = "replicate"
)
result
```

---

get_top_n_genes *Get Top Genes From An AssociationTestResult*

Description

Pulls the genes with the highest wald statistic from an association test result, with a p value cutoff.

Usage

```
get_top_n_genes(
  association_test_results,
  n_genes = 40,
  lineage = NA,
  p_cutoff = 0.05
)
```

Arguments

association_test_results	Dataframe. The association test results data frame to take the genes from. Generated by tradeSeq::associationTest.
n_genes	Integer. The number of genes to return. Defaults to 40.
lineage	The Lineage to use. The Defaults to NA, which assumes the test was run with Lineages=False.
p_cutoff	Decimal. The maximum P value cutoff to use. Defaults to 0.05.

Value

A vector of strings. The names of the genes that best describe a lineage's trajectory.

Examples

```
assoRes <- data.frame(
    row.names = c("A", "B", "C", "D"),
    waldStat = c(25, 50, 100, 10),
    pvalue = c(0.01, 0.5, 0.005, 0.13)
)
get_top_n_genes(assoRes, n_genes = 2)
```

MappingResult *Blase Mapping Result*

Description

Created by `map_best_bin()`

Usage

```
MappingResult(
  bulk_name,
  best_bin,
  best_correlation,
  top_2_distance,
  strong_mapping,
  history,
  bootstrap_iterations,
  metric = "spearman"
)
```

Arguments

`bulk_name`	String. The name of the bulk sample being mapped.
`best_bin`	Integer. The bin that best matched the bulk sample.
`best_correlation`	Decimal. The spearman's rho that the test geneset had between the winning bin and the bulk.
`top_2_distance`	Decimal. The absolute difference between the best and second best mapping buckets. Higher indicates a less doubtful mapping.
`strong_mapping`	Boolean. TRUE when the mapped bin's lower bound is higher than the maximum upper bound of the other bins.
`history`	A dataframe of the correlation score (decimal) and confidence bounds (decimal pairs) for each bin. Access with `mapping_history()`
`bootstrap_iterations`	Integer. The number of iterations used during the bootstrap.
`metric`	Character. The metric used to evaluate mappings.

Value

A MappingResult object

See Also

map_best_bin()

Examples

```
counts_matrix <- matrix(
    c(seq_len(120) / 10, seq_len(120) / 5),
    ncol = 48, nrow = 5
)
sce <- SingleCellExperiment::SingleCellExperiment(assays = list(
    normcounts = counts_matrix, logcounts = log(counts_matrix)
))
colnames(sce) <- seq_len(48)
rownames(sce) <- as.character(seq_len(5))
sce$cell_type <- c(rep("celltype_1", 24), rep("celltype_2", 24))

sce$pseudotime <- seq_len(48) - 1
blase_data <- as.BlaseData(sce, pseudotime_slot = "pseudotime", n_bins = 4)
genes(blase_data) <- as.character(seq_len(5))

bulk_counts <- matrix(seq_len(15) * 10, ncol = 3, nrow = 5)
colnames(bulk_counts) <- c("A", "B", "C")
rownames(bulk_counts) <- as.character(seq_len(5))

# Map to bin
result <- map_best_bin(blase_data, "B", bulk_counts)
result

# Map all bulks to bin
results <- map_all_best_bins(blase_data, bulk_counts)

# Plot Heatmap
plot_mapping_result_heatmap(list(result))

# Plot Correlation
plot_mapping_result_corr(result)

# Plot populations
sce <- assign_pseudotime_bins(
    sce,
    pseudotime_slot = "pseudotime", n_bins = 4
)
plot_bin_population(sce, best_bin(result), group_by_slot = "cell_type")

# Getters
bulk_name(result)
best_bin(result)
best_correlation(result)
```

```
top_2_distance(result)
strong_mapping(result)
mapping_history(result)
bootstrap_iterations(result)
metric(result)

# Setters
bulk_name(result) <- "New Name"
```

`mapping_history` *Get the mapping history for a BLASE Mapping Results object.*

Description

Get the mapping history for a BLASE Mapping Results object.

Usage

```
mapping_history(x)

## S4 method for signature 'MappingResult'
mapping_history(x)
```

Arguments

x	a MappingResult object

Value

The mapping history of this mapping, in a data frame.

Examples

```
counts_matrix <- matrix(
    c(seq_len(120) / 10, seq_len(120) / 5),
    ncol = 48, nrow = 5
)
sce <- SingleCellExperiment::SingleCellExperiment(assays = list(
    normcounts = counts_matrix, logcounts = log(counts_matrix)
))
colnames(sce) <- seq_len(48)
rownames(sce) <- as.character(seq_len(5))
sce$cell_type <- c(rep("celltype_1", 24), rep("celltype_2", 24))

sce$pseudotime <- seq_len(48) - 1
blase_data <- as.BlaseData(sce, pseudotime_slot = "pseudotime", n_bins = 4)
genes(blase_data) <- as.character(seq_len(5))

bulk_counts <- matrix(seq_len(15) * 10, ncol = 3, nrow = 5)
colnames(bulk_counts) <- c("A", "B", "C")
```

```
rownames(bulk_counts) <- as.character(seq_len(5))

# Map to bin
result <- map_best_bin(blase_data, "B", bulk_counts)
result

# Map all bulks to bin
results <- map_all_best_bins(blase_data, bulk_counts)

# Plot Heatmap
plot_mapping_result_heatmap(list(result))

# Plot Correlation
plot_mapping_result_corr(result)

# Plot populations
sce <- assign_pseudotime_bins(
    sce,
    pseudotime_slot = "pseudotime", n_bins = 4
)
plot_bin_population(sce, best_bin(result), group_by_slot = "cell_type")

# Getters
bulk_name(result)
best_bin(result)
best_correlation(result)
top_2_distance(result)
strong_mapping(result)
mapping_history(result)
bootstrap_iterations(result)
metric(result)

# Setters
bulk_name(result) <- "New Name"
```

`map_all_best_bins` *Map many bulk samples in the same dataframe*

Description

Map many bulk samples in the same dataframe

Usage

```
map_all_best_bins(
  blase_data,
  bulk_data,
  bootstrap_iterations = 200,
  confidence_level = 0.9,
  BPPARAM = BiocParallel::SerialParam(),
```

```
  metric = "spearman"
)
```

Arguments

blase_data	The BlaseData holding the bins and pseudobulks.
bulk_data	Dataframe. The whole bulk read matrix as a dataframe. Each row should represent a gene, and each column a sample.
bootstrap_iterations	Integer. The number of bootstrapping iterations to run.
confidence_level	Decimal between 0-1. The confidence interval to calculate for mappings. Defaults to 0.9, or 90%.
BPPARAM	The BiocParallel::BiocParallelParam for multithreading if desired. Defaults to `BiocParallel::SerialParam()`
metric	Character. The metric to use to compare mappings. One of: 'spearman', 'pearson', 'kendall', 'cosine_similarity', 'euclidean', 'manhattan.'

Value

A vector of MappingResult objects.

See Also

`map_best_bin()`

Examples

```
counts_matrix <- matrix(
    c(seq_len(120) / 10, seq_len(120) / 5),
    ncol = 48, nrow = 5
)
sce <- SingleCellExperiment::SingleCellExperiment(assays = list(
    normcounts = counts_matrix, logcounts = log(counts_matrix)
))
colnames(sce) <- seq_len(48)
rownames(sce) <- as.character(seq_len(5))
sce$cell_type <- c(rep("celltype_1", 24), rep("celltype_2", 24))

sce$pseudotime <- seq_len(48) - 1
blase_data <- as.BlaseData(sce, pseudotime_slot = "pseudotime", n_bins = 4)
genes(blase_data) <- as.character(seq_len(5))

bulk_counts <- matrix(seq_len(15) * 10, ncol = 3, nrow = 5)
colnames(bulk_counts) <- c("A", "B", "C")
rownames(bulk_counts) <- as.character(seq_len(5))

# Map to bin
result <- map_best_bin(blase_data, "B", bulk_counts)
result
```

```
# Map all bulks to bin
results <- map_all_best_bins(blase_data, bulk_counts)

# Plot Heatmap
plot_mapping_result_heatmap(list(result))

# Plot Correlation
plot_mapping_result_corr(result)

# Plot populations
sce <- assign_pseudotime_bins(
    sce,
    pseudotime_slot = "pseudotime", n_bins = 4
)
plot_bin_population(sce, best_bin(result), group_by_slot = "cell_type")

# Getters
bulk_name(result)
best_bin(result)
best_correlation(result)
top_2_distance(result)
strong_mapping(result)
mapping_history(result)
bootstrap_iterations(result)
metric(result)

# Setters
bulk_name(result) <- "New Name"
```

`map_best_bin` *Map the best matching SC bin for a bulk sample*

Description

Map the best matching SC bin for a bulk sample

Usage

```
map_best_bin(
  blase_data,
  bulk_id,
  bulk_data,
  bootstrap_iterations = 200,
  confidence_level = 0.9,
  metric = "spearman",
  log_data = FALSE
)
```

Arguments

blase_data	The BlaseData holding the bins.
bulk_id	String. The sample id of the bulk to analyse.
bulk_data	Dataframe. The whole bulk read matrix as a dataframe. Each row should represent a gene, and each column a sample.
bootstrap_iterations	Integer. The number of bootstrapping iterations to run.
confidence_level	Decimal between 0-1. The confidence interval to calculate for mappings. Defaults to 90%.
metric	Character. The metric to use to compare mappings. One of: 'spearman', 'pearson', 'kendall', 'cosine_similarity', 'euclidean', 'manhattan.'
log_data	Boolean. When true, bulk and bin values are log2 transformed

Value

A MappingResult object.

Examples

```
counts_matrix <- matrix(
    c(seq_len(120) / 10, seq_len(120) / 5),
    ncol = 48, nrow = 5
)
sce <- SingleCellExperiment::SingleCellExperiment(assays = list(
    normcounts = counts_matrix, logcounts = log(counts_matrix)
))
colnames(sce) <- seq_len(48)
rownames(sce) <- as.character(seq_len(5))
sce$cell_type <- c(rep("celltype_1", 24), rep("celltype_2", 24))

sce$pseudotime <- seq_len(48) - 1
blase_data <- as.BlaseData(sce, pseudotime_slot = "pseudotime", n_bins = 4)
genes(blase_data) <- as.character(seq_len(5))

bulk_counts <- matrix(seq_len(15) * 10, ncol = 3, nrow = 5)
colnames(bulk_counts) <- c("A", "B", "C")
rownames(bulk_counts) <- as.character(seq_len(5))

# Map to bin
result <- map_best_bin(blase_data, "B", bulk_counts)
result

# Map all bulks to bin
results <- map_all_best_bins(blase_data, bulk_counts)

# Plot Heatmap
plot_mapping_result_heatmap(list(result))
```

```
# Plot Correlation
plot_mapping_result_corr(result)

# Plot populations
sce <- assign_pseudotime_bins(
    sce,
    pseudotime_slot = "pseudotime", n_bins = 4
)
plot_bin_population(sce, best_bin(result), group_by_slot = "cell_type")

# Getters
bulk_name(result)
best_bin(result)
best_correlation(result)
top_2_distance(result)
strong_mapping(result)
mapping_history(result)
bootstrap_iterations(result)
metric(result)

# Setters
bulk_name(result) <- "New Name"
```

MCA_PF_SCE *Malaria Cell Atlas Plasmodium falciparum for BLASE Vignette*

Description

Data from the Malaria Cell Atlas, with the following additional processing:

1. Genes renamed to match bulk samples in vignette
2. Subset to 2500 cells
3. Normalised
4. Highly variable genes identified
5. Pseudotime calculated
6. Genes subset to include a spread of those found to have high ratios by BLASE's "Gene Peakedness" measure.

Usage

```
MCA_PF_SCE
```

Format

An object of class `SingleCellExperiment` with 1746 rows and 2500 columns.

Source

https://www.malariacellatlas.org/atlas/plasmodium-falciparum-atlas/

metric *Get the mapping history for a BLASE Mapping Results object.*

Description

Get the mapping history for a BLASE Mapping Results object.

Usage

```
metric(x)

## S4 method for signature 'MappingResult'
metric(x)
```

Arguments

x	a MappingResult object

Value

a String, the metric used to calculate the result.

Examples

```
counts_matrix <- matrix(
    c(seq_len(120) / 10, seq_len(120) / 5),
    ncol = 48, nrow = 5
)
sce <- SingleCellExperiment::SingleCellExperiment(assays = list(
    normcounts = counts_matrix, logcounts = log(counts_matrix)
))
colnames(sce) <- seq_len(48)
rownames(sce) <- as.character(seq_len(5))
sce$cell_type <- c(rep("celltype_1", 24), rep("celltype_2", 24))

sce$pseudotime <- seq_len(48) - 1
blase_data <- as.BlaseData(sce, pseudotime_slot = "pseudotime", n_bins = 4)
genes(blase_data) <- as.character(seq_len(5))

bulk_counts <- matrix(seq_len(15) * 10, ncol = 3, nrow = 5)
colnames(bulk_counts) <- c("A", "B", "C")
rownames(bulk_counts) <- as.character(seq_len(5))

# Map to bin
result <- map_best_bin(blase_data, "B", bulk_counts)
result

# Map all bulks to bin
results <- map_all_best_bins(blase_data, bulk_counts)
```

```
# Plot Heatmap
plot_mapping_result_heatmap(list(result))

# Plot Correlation
plot_mapping_result_corr(result)

# Plot populations
sce <- assign_pseudotime_bins(
    sce,
    pseudotime_slot = "pseudotime", n_bins = 4
)
plot_bin_population(sce, best_bin(result), group_by_slot = "cell_type")

# Getters
bulk_name(result)
best_bin(result)
best_correlation(result)
top_2_distance(result)
strong_mapping(result)
mapping_history(result)
bootstrap_iterations(result)
metric(result)

# Setters
bulk_name(result) <- "New Name"
```

`painter_microarray` *Painter 2018 Plasmodium falciparum 48h asexual lifecycle microarray data*

Description

Data originally from https://doi.org/10.1038/s41467-018-04966-3. Used as generated in the BLASE reproducibility documents available at https://zenodo.org/records/16615703, however genes have been subset to reduce file size.

Usage

```
painter_microarray
```

Format

An object of class `data.frame` with 1731 rows and 48 columns.

Source

https://zenodo.org/records/16615703

`plot_bin_population` *Plot the populations of a bin*

Description

Plot the populations of a bin

Usage

```
plot_bin_population(x, bin, ...)

## S4 method for signature 'SingleCellExperiment'
plot_bin_population(x, bin, group_by_slot)
```

Arguments

`x`	An object to plot on.
`bin`	Integer. The pseudotime bin to plot
`...`	additional arguments passed to object-specific methods.
`group_by_slot`	String. The metadata column in the SingleCellExperiment::SingleCellExperiment to be used as the cell type labels.

Value

A ggplot2 object of a plot of population in the given object for this bin.

Examples

```
counts_matrix <- matrix(
    c(seq_len(120) / 10, seq_len(120) / 5),
    ncol = 48, nrow = 5
)
sce <- SingleCellExperiment::SingleCellExperiment(assays = list(
    normcounts = counts_matrix, logcounts = log(counts_matrix)
))
colnames(sce) <- seq_len(48)
rownames(sce) <- as.character(seq_len(5))
sce$cell_type <- c(rep("celltype_1", 24), rep("celltype_2", 24))

sce$pseudotime <- seq_len(48) - 1
blase_data <- as.BlaseData(sce, pseudotime_slot = "pseudotime", n_bins = 4)
genes(blase_data) <- as.character(seq_len(5))

bulk_counts <- matrix(seq_len(15) * 10, ncol = 3, nrow = 5)
colnames(bulk_counts) <- c("A", "B", "C")
rownames(bulk_counts) <- as.character(seq_len(5))

# Map to bin
```

```
result <- map_best_bin(blase_data, "B", bulk_counts)
result

# Map all bulks to bin
results <- map_all_best_bins(blase_data, bulk_counts)

# Plot Heatmap
plot_mapping_result_heatmap(list(result))

# Plot Correlation
plot_mapping_result_corr(result)

# Plot populations
sce <- assign_pseudotime_bins(
    sce,
    pseudotime_slot = "pseudotime", n_bins = 4
)
plot_bin_population(sce, best_bin(result), group_by_slot = "cell_type")

# Getters
bulk_name(result)
best_bin(result)
best_correlation(result)
top_2_distance(result)
strong_mapping(result)
mapping_history(result)
bootstrap_iterations(result)
metric(result)

# Setters
bulk_name(result) <- "New Name"
```

`plot_find_best_params_results` *Plot the results of the search for good parameters*

Description

Plot the results of the search for good parameters

Usage

```
plot_find_best_params_results(
  find_best_params_results,
  bin_count_colors = viridis::scale_color_viridis(option = "viridis"),
  gene_count_colors = viridis::scale_color_viridis(option = "magma")
)
```

Arguments

find_best_params_results
: Dataframe. Results dataframe from find_best_params()

bin_count_colors
: Optional, custom bin count scale color scheme.

gene_count_colors
: Optional, custom gene count scale color scheme.

Value

A plot showing how convexity changes as n_bins and n_genes are changed. See find_best_params() for details on how to interpret.

See Also

find_best_params()

Examples

```
ncells <- 70
ngenes <- 100
counts_matrix <- matrix(
    c(seq_len(3500) / 10, seq_len(3500) / 5),
    ncol = ncells,
    nrow = ngenes
)
sce <- SingleCellExperiment::SingleCellExperiment(assays = list(
    normcounts = counts_matrix, logcounts = log(counts_matrix)
))
colnames(sce) <- paste0("cell", seq_len(ncells))
rownames(sce) <- paste0("gene", seq_len(ngenes))
sce$cell_type <- c(
    rep("celltype_1", ncells / 2),
    rep("celltype_2", ncells / 2)
)

sce$pseudotime <- seq_len(ncells) - 1
genelist <- rownames(sce)

# Finding the best params for the BlaseData
best_params <- find_best_params(
    sce, genelist,
    bins_count_range = c(2, 3),
    gene_count_range = c(20, 50),
    pseudotime_slot = "pseudotime",
    split_by = "pseudotime_range"
)
best_params
plot_find_best_params_results(best_params)
```

plot_gene_peakedness *plot_gene_peakedness*

Description

plot_gene_peakedness

Usage

```
plot_gene_peakedness(
  sce,
  gene_peakedness_df,
  gene,
  pseudotime_slot = "slingPseudotime_1"
)
```

Arguments

sce	SingleCellExperiment::SingleCellExperiment to plot gene from. Must contain pseudotime, and normcounts
gene_peakedness_df	The DataFrame Result of `calculate_gene_peakedness`
gene	String. The name of the gene to plot. Must be present in the SCE and gene_peakedness_df
pseudotime_slot	String. The pseudotime column in the SingleCellExperiment::SingleCellExperiment object metadata.

Value

A ggplot2::ggplot2 plot showing: in black points, expression of the gene over pseudotime, in a green line, the fitted expression of the gene over pseudotime, the inside and outside of window means of smoothed expression (red and blue dotted horizotal lines respectively), and the bounds of the window (in black dotted vertical lines).

Examples

```
ncells <- 70
ngenes <- 100
# Each gene should have mean around its gene number
counts <- c()
for (i in seq_len(ngenes)) {
    counts <- c(counts, dnorm(seq_len(ncells), mean = (ncells / i), sd = 1))
}

counts_matrix <- matrix(
    counts,
    ncol = ncells,
    nrow = ngenes
```

```
)
sce <- SingleCellExperiment::SingleCellExperiment(assays = list(
    counts = counts_matrix * 3,
    normcounts = counts_matrix,
    logcounts = log(counts_matrix)
))
colnames(sce) <- paste0("cell", seq_len(ncells))
rownames(sce) <- paste0("gene", seq_len(ngenes))
sce$cell_type <- c(
    rep("celltype_1", ncells / 2),
    rep("celltype_2", ncells / 2)
)

sce$pseudotime <- seq_len(ncells) - 1
genelist <- rownames(sce)

# calculate_gene_peakedness
gene_peakedness <- calculate_gene_peakedness(
    sce,
    pseudotime_slot = "pseudotime"
)

head(gene_peakedness)

# plot_gene_peakedness
plot_gene_peakedness(sce, gene_peakedness, "gene20",
    pseudotime_slot = "pseudotime"
)

# smooth_gene
smoothed_gene20 <- smooth_gene(
    sce, "gene20",
    pseudotime_slot = "pseudotime"
)
head(smoothed_gene20)

# Select best spread of genes
genes_to_use <- gene_peakedness_spread_selection(sce, gene_peakedness,
    genes_per_bin = 2, n_gene_bins = 1, pseudotime_slot = "pseudotime"
)

print(genes_to_use)
plot(
    x = gene_peakedness[
        gene_peakedness$gene %in% genes_to_use, "peak_pseudotime"
    ],
    y = gene_peakedness[gene_peakedness$gene %in% genes_to_use, "ratio"]
)
```

---

plot_mapping_result *Plot a summary of the mapping result*

Description

Plot a summary of the mapping result

Usage

```
plot_mapping_result(x, y, ...)

## S4 method for signature 'SingleCellExperiment,MappingResult'
plot_mapping_result(x, y, group_by_slot)
```

Arguments

x	An object to plot on.
y	The MappingResult object to plot
...	additional arguments passed to object-specific methods.
group_by_slot	String. The metadata column in the SingleCellExperiment::SingleCellExperiment to be used as the coloring for the output plot. Passed to `scater::plotUMAP()` as `colour_by`, and will be used to produce a bar chart of populations in the best mapped bin.

Value

A set of plots describing the mapping.

See Also

`plot_mapping_result_corr()`, `plot_bin_population()`

Examples

```
counts_matrix <- matrix(
    c(seq_len(120) / 10, seq_len(120) / 5),
    ncol = 48, nrow = 5
)
sce <- SingleCellExperiment::SingleCellExperiment(assays = list(
    normcounts = counts_matrix, logcounts = log(counts_matrix)
))
colnames(sce) <- seq_len(48)
rownames(sce) <- as.character(seq_len(5))
sce$cell_type <- c(rep("celltype_1", 24), rep("celltype_2", 24))

sce$pseudotime <- seq_len(48) - 1
blase_data <- as.BlaseData(sce, pseudotime_slot = "pseudotime", n_bins = 4)
genes(blase_data) <- as.character(seq_len(5))

bulk_counts <- matrix(seq_len(15) * 10, ncol = 3, nrow = 5)
colnames(bulk_counts) <- c("A", "B", "C")
rownames(bulk_counts) <- as.character(seq_len(5))

result <- map_best_bin(blase_data, "B", bulk_counts)
```

```
# Plot bin
sce <- scater::runUMAP(sce)
sce <- assign_pseudotime_bins(
    sce,
    pseudotime_slot = "pseudotime", n_bins = 4
)
plot_mapping_result(sce, result, group_by_slot = "cell_type")
```

`plot_mapping_result_corr` *Plot a mapping result's correlation*

Description

Plots the mapping results correlations with each pseudotime bin

Usage

```
plot_mapping_result_corr(mapping_result)
```

Arguments

`mapping_result` A MappingResult object to plot the correlations for.

Value

A ggplot2::ggplot2 object of the the line plot

Examples

```
counts_matrix <- matrix(
    c(seq_len(120) / 10, seq_len(120) / 5),
    ncol = 48, nrow = 5
)
sce <- SingleCellExperiment::SingleCellExperiment(assays = list(
    normcounts = counts_matrix, logcounts = log(counts_matrix)
))
colnames(sce) <- seq_len(48)
rownames(sce) <- as.character(seq_len(5))
sce$cell_type <- c(rep("celltype_1", 24), rep("celltype_2", 24))

sce$pseudotime <- seq_len(48) - 1
blase_data <- as.BlaseData(sce, pseudotime_slot = "pseudotime", n_bins = 4)
genes(blase_data) <- as.character(seq_len(5))

bulk_counts <- matrix(seq_len(15) * 10, ncol = 3, nrow = 5)
colnames(bulk_counts) <- c("A", "B", "C")
rownames(bulk_counts) <- as.character(seq_len(5))
```

```
# Map to bin
result <- map_best_bin(blase_data, "B", bulk_counts)
result

# Map all bulks to bin
results <- map_all_best_bins(blase_data, bulk_counts)

# Plot Heatmap
plot_mapping_result_heatmap(list(result))

# Plot Correlation
plot_mapping_result_corr(result)

# Plot populations
sce <- assign_pseudotime_bins(
    sce,
    pseudotime_slot = "pseudotime", n_bins = 4
)
plot_bin_population(sce, best_bin(result), group_by_slot = "cell_type")

# Getters
bulk_name(result)
best_bin(result)
best_correlation(result)
top_2_distance(result)
strong_mapping(result)
mapping_history(result)
bootstrap_iterations(result)
metric(result)

# Setters
bulk_name(result) <- "New Name"
```

`plot_mapping_result_heatmap` *Plot a mapping result heatmap*

Description

Plots Spearman's Rho as the fill colour, and adds * if the MappingResult was strongly assigned.

Usage

```
plot_mapping_result_heatmap(
  mapping_result_list,
  heatmap_fill_scale = NULL,
  annotate_strong = TRUE,
  annotate_correlation = FALSE,
  bin_order = NULL,
```

```
  text_background = FALSE
)
```

Arguments

mapping_result_list	A list of MappingResult objects to include in the heatmap.
heatmap_fill_scale	The ggplot2 compatible fill gradient scale to apply to the heatmap.
annotate_strong	Boolan. Whether to annotate the heatmap with strong results or not, defaults to TRUE.
annotate_correlation	Boolean. Whether to annotate the heatmap with the correlation of bin to each bulk sample. Defaults to FALSE.
bin_order	Vector of integers. A vector of the bin ids in which to plot the pseudotime bins along the x-axis.
text_background	Boolean. Whether to show background on labels or not. Has no effect if no annotations are enabled.

Value

A ggplot2::ggplot2 heatmap showing the correlations of each mapping result across every pseudotime bin.

Examples

```
counts_matrix <- matrix(
    c(seq_len(120) / 10, seq_len(120) / 5),
    ncol = 48, nrow = 5
)
sce <- SingleCellExperiment::SingleCellExperiment(assays = list(
    normcounts = counts_matrix, logcounts = log(counts_matrix)
))
colnames(sce) <- seq_len(48)
rownames(sce) <- as.character(seq_len(5))
sce$cell_type <- c(rep("celltype_1", 24), rep("celltype_2", 24))

sce$pseudotime <- seq_len(48) - 1
blase_data <- as.BlaseData(sce, pseudotime_slot = "pseudotime", n_bins = 4)
genes(blase_data) <- as.character(seq_len(5))

bulk_counts <- matrix(seq_len(15) * 10, ncol = 3, nrow = 5)
colnames(bulk_counts) <- c("A", "B", "C")
rownames(bulk_counts) <- as.character(seq_len(5))

# Map to bin
result <- map_best_bin(blase_data, "B", bulk_counts)
result
```

```
# Map all bulks to bin
results <- map_all_best_bins(blase_data, bulk_counts)

# Plot Heatmap
plot_mapping_result_heatmap(list(result))

# Plot Correlation
plot_mapping_result_corr(result)

# Plot populations
sce <- assign_pseudotime_bins(
    sce,
    pseudotime_slot = "pseudotime", n_bins = 4
)
plot_bin_population(sce, best_bin(result), group_by_slot = "cell_type")

# Getters
bulk_name(result)
best_bin(result)
best_correlation(result)
top_2_distance(result)
strong_mapping(result)
mapping_history(result)
bootstrap_iterations(result)
metric(result)

# Setters
bulk_name(result) <- "New Name"
```

PRIVATE_.ci *.ci*

Description

Originally implemented in RVAidemoire Version 0.9-83-7.

Usage

```
PRIVATE_.ci(x, conf.level = 0.95)
```

Arguments

x	data to calculate ci for
conf.level	confidence level to calculate

Value

confidence interval results

`PRIVATE_correlation.ci` *Confidence interval of a correlation coefficient*

Description

Computes the confidence interval of a correlation coefficient by bootstraping. Adapted from the implementation of spearman.ci in RVAidemoire Version 0.9-83-7.

Usage

```
PRIVATE_correlation.ci(
  var1,
  var2,
  nrep = 1000,
  conf.level = 0.95,
  metric = "spearman"
)
```

Arguments

`var1`	numeric vector (first variable).
`var2`	nuermic verctor (second variable).
`nrep`	number of replicates for bootstrapping.
`conf.level`	confidence level of the interval.
`metric`	character from "pearson", "spearman", "kendall", the correlation method to use.

Value

description method name of the test.

data.name a character string giving the name(s) of the data.

conf.level confidence level.

rep number of replicates.

estimate of correlation coefficient.

conf.int confidence interval.

PRIVATE_cosine_similarity.ci *Confidence interval of cosine similarity*

Description

Computes the confidence interval of a cosine similarity coefficient by bootstraping. Adapted from the implementation of spearman.ci in RVAidemoire Version 0.9-83-7.

Usage

```
PRIVATE_cosine_similarity.ci(var1, var2, nrep = 1000, conf.level = 0.95)
```

Arguments

var1	numeric vector (first variable).
var2	nuermic verctor (second variable).
nrep	number of replicates for bootstrapping.
conf.level	confidence level of the interval.

Value

description method name of the test.

data.name a character string giving the name(s) of the data.

conf.level confidence level.

rep number of replicates.

estimate Spearman's rank correlation coefficient.

conf.int confidence interval.

PRIVATE_distance.ci *Confidence interval of a distance metric*

Description

Computes the confidence interval of a Spearman's rank correlation coefficient by bootstraping. Adapted from the implementation of spearman.ci in RVAidemoire Version 0.9-83-7.

Usage

```
PRIVATE_distance.ci(
  var1,
  var2,
  nrep = 1000,
  conf.level = 0.95,
  metric = "euclidean"
)
```

Arguments

var1	numeric vector (first variable).
var2	nuermic verctor (second variable).
nrep	number of replicates for bootstrapping.
conf.level	confidence level of the interval.
metric	character from "euclidean", "manhattan", the distance method to use.

Value

description method name of the test.

data.name a character string giving the name(s) of the data.

conf.level confidence level.

rep number of replicates.

estimate calculated distance (as a negative, so that there is consistency with other methods where a higher value indicates more similarity).

conf.int confidence interval.

---

pseudobulk_bins *Get pseudobulk bins of a BLASE Data object.*

Description

Get pseudobulk bins of a BLASE Data object.

Usage

```
pseudobulk_bins(x)

## S4 method for signature 'BlaseData'
pseudobulk_bins(x)
```

Arguments

x	a BlaseData object

Value

List of dataframes. Each dataframe is the normalised counts of cells by genes in each pseudotime bin. List index is the pseudotime bin.

Examples

```
counts <- matrix(rpois(100, lambda = 10), ncol = 10, nrow = 10)
sce <- SingleCellExperiment::SingleCellExperiment(
    assays = list(normcounts = counts)
)
sce$pseudotime <- seq_len(10) - 1
data <- as.BlaseData(sce, pseudotime_slot = "pseudotime", n_bins = 3)
genes(data) <- as.character(seq_len(10))

genes(data)
```

---

pseudobulk_bins<- *Set genes of a BLASE Data object.*

Description

Set genes of a BLASE Data object.

Usage

```
pseudobulk_bins(x) <- value

## S4 replacement method for signature 'BlaseData'
pseudobulk_bins(x) <- value
```

Arguments

x	a BlaseData object
value	List of dataframes. Each dataframe is the normalised counts of cells by genes in each pseudotime bin. List index is the pseudotime bin.

Value

Nothing

---

show,BlaseData-method *Show an BlaseData object*

Description

Show an BlaseData object

Usage

```
## S4 method for signature 'BlaseData'
show(object)
```

Arguments

object	a BlaseData object

Value

A character vector describing the BLASE object

Examples

```
counts <- matrix(rpois(100, lambda = 10), ncol = 10, nrow = 10)
sce <- SingleCellExperiment::SingleCellExperiment(
    assays = list(normcounts = counts)
)
sce$pseudotime <- seq_len(10) - 1
data <- as.BlaseData(sce, pseudotime_slot = "pseudotime", n_bins = 3)
genes(data) <- as.character(seq_len(10))

genes(data)
```

---

show,MappingResult-method

Show an MappingResult object

Description

Show an MappingResult object

Usage

```
## S4 method for signature 'MappingResult'
show(object)
```

Arguments

object	an MappingResult object

Value

A character vector describing the Mapping Result object

Examples

```
counts_matrix <- matrix(
    c(seq_len(120) / 10, seq_len(120) / 5),
    ncol = 48, nrow = 5
)
sce <- SingleCellExperiment::SingleCellExperiment(assays = list(
    normcounts = counts_matrix, logcounts = log(counts_matrix)
))
```

```
colnames(sce) <- seq_len(48)
rownames(sce) <- as.character(seq_len(5))
sce$cell_type <- c(rep("celltype_1", 24), rep("celltype_2", 24))

sce$pseudotime <- seq_len(48) - 1
blase_data <- as.BlaseData(sce, pseudotime_slot = "pseudotime", n_bins = 4)
genes(blase_data) <- as.character(seq_len(5))

bulk_counts <- matrix(seq_len(15) * 10, ncol = 3, nrow = 5)
colnames(bulk_counts) <- c("A", "B", "C")
rownames(bulk_counts) <- as.character(seq_len(5))

# Map to bin
result <- map_best_bin(blase_data, "B", bulk_counts)
result

# Map all bulks to bin
results <- map_all_best_bins(blase_data, bulk_counts)

# Plot Heatmap
plot_mapping_result_heatmap(list(result))

# Plot Correlation
plot_mapping_result_corr(result)

# Plot populations
sce <- assign_pseudotime_bins(
    sce,
    pseudotime_slot = "pseudotime", n_bins = 4
)
plot_bin_population(sce, best_bin(result), group_by_slot = "cell_type")

# Getters
bulk_name(result)
best_bin(result)
best_correlation(result)
top_2_distance(result)
strong_mapping(result)
mapping_history(result)
bootstrap_iterations(result)
metric(result)

# Setters
bulk_name(result) <- "New Name"
```

---

smooth_gene *smooth_gene*

Description

Returns the smoothed expression of the given gene, based on a GAM fit to the normalised expression.

Usage

```
smooth_gene(sce, gene, pseudotime_slot = "slingPseudotime_1", knots = 10)
```

Arguments

sce	SingleCellExperiment::SingleCellExperiment to do the calculations on.
gene	String. The name of the gene to smooth
pseudotime_slot	String. The slot in the SingleCellExperiment::SingleCellExperiment object metadata containing pseudotime
knots	Integer. The number of knots to use when fitting the GAM

Value

Smoothed Gene Expression over pseudotime

Examples

```
ncells <- 70
ngenes <- 100
# Each gene should have mean around its gene number
counts <- c()
for (i in seq_len(ngenes)) {
    counts <- c(counts, dnorm(seq_len(ncells), mean = (ncells / i), sd = 1))
}

counts_matrix <- matrix(
    counts,
    ncol = ncells,
    nrow = ngenes
)
sce <- SingleCellExperiment::SingleCellExperiment(assays = list(
    counts = counts_matrix * 3,
    normcounts = counts_matrix,
    logcounts = log(counts_matrix)
))
colnames(sce) <- paste0("cell", seq_len(ncells))
rownames(sce) <- paste0("gene", seq_len(ngenes))
sce$cell_type <- c(
    rep("celltype_1", ncells / 2),
    rep("celltype_2", ncells / 2)
)

sce$pseudotime <- seq_len(ncells) - 1
genelist <- rownames(sce)
```

```
# calculate_gene_peakedness
gene_peakedness <- calculate_gene_peakedness(
    sce,
    pseudotime_slot = "pseudotime"
)

head(gene_peakedness)

# plot_gene_peakedness
plot_gene_peakedness(sce, gene_peakedness, "gene20",
    pseudotime_slot = "pseudotime"
)

# smooth_gene
smoothed_gene20 <- smooth_gene(
    sce, "gene20",
    pseudotime_slot = "pseudotime"
)
head(smoothed_gene20)

# Select best spread of genes
genes_to_use <- gene_peakedness_spread_selection(sce, gene_peakedness,
    genes_per_bin = 2, n_gene_bins = 1, pseudotime_slot = "pseudotime"
)

print(genes_to_use)
plot(
    x = gene_peakedness[
        gene_peakedness$gene %in% genes_to_use, "peak_pseudotime"
    ],
    y = gene_peakedness[gene_peakedness$gene %in% genes_to_use, "ratio"]
)
```

`strong_mapping` *Get if the result is strong for a BLASE Mapping Results object.*

Description

Get if the result is strong for a BLASE Mapping Results object.

Usage

```
strong_mapping(x)

## S4 method for signature 'MappingResult'
strong_mapping(x)
```

Arguments

x	a MappingResult object

Value

Boolean. TRUE if the result is strong, otherwise FALSE

Examples

```
counts_matrix <- matrix(
    c(seq_len(120) / 10, seq_len(120) / 5),
    ncol = 48, nrow = 5
)
sce <- SingleCellExperiment::SingleCellExperiment(assays = list(
    normcounts = counts_matrix, logcounts = log(counts_matrix)
))
colnames(sce) <- seq_len(48)
rownames(sce) <- as.character(seq_len(5))
sce$cell_type <- c(rep("celltype_1", 24), rep("celltype_2", 24))

sce$pseudotime <- seq_len(48) - 1
blase_data <- as.BlaseData(sce, pseudotime_slot = "pseudotime", n_bins = 4)
genes(blase_data) <- as.character(seq_len(5))

bulk_counts <- matrix(seq_len(15) * 10, ncol = 3, nrow = 5)
colnames(bulk_counts) <- c("A", "B", "C")
rownames(bulk_counts) <- as.character(seq_len(5))

# Map to bin
result <- map_best_bin(blase_data, "B", bulk_counts)
result

# Map all bulks to bin
results <- map_all_best_bins(blase_data, bulk_counts)

# Plot Heatmap
plot_mapping_result_heatmap(list(result))

# Plot Correlation
plot_mapping_result_corr(result)

# Plot populations
sce <- assign_pseudotime_bins(
    sce,
    pseudotime_slot = "pseudotime", n_bins = 4
)
plot_bin_population(sce, best_bin(result), group_by_slot = "cell_type")

# Getters
bulk_name(result)
best_bin(result)
best_correlation(result)
```

```
top_2_distance(result)
strong_mapping(result)
mapping_history(result)
bootstrap_iterations(result)
metric(result)

# Setters
bulk_name(result) <- "New Name"
```

`top_2_distance` *Get the difference in correlation between the top 2 most correlated bins for a BLASE Mapping Results object.*

Description

Get the difference in correlation between the top 2 most correlated bins for a BLASE Mapping Results object.

Usage

```
top_2_distance(x)

## S4 method for signature 'MappingResult'
top_2_distance(x)
```

Arguments

x	a MappingResult object

Value

Decimal. The difference in correlation between the top 2 most correlated bins for this mapping.

Examples

```
counts_matrix <- matrix(
    c(seq_len(120) / 10, seq_len(120) / 5),
    ncol = 48, nrow = 5
)
sce <- SingleCellExperiment::SingleCellExperiment(assays = list(
    normcounts = counts_matrix, logcounts = log(counts_matrix)
))
colnames(sce) <- seq_len(48)
rownames(sce) <- as.character(seq_len(5))
sce$cell_type <- c(rep("celltype_1", 24), rep("celltype_2", 24))

sce$pseudotime <- seq_len(48) - 1
blase_data <- as.BlaseData(sce, pseudotime_slot = "pseudotime", n_bins = 4)
genes(blase_data) <- as.character(seq_len(5))
```

```
bulk_counts <- matrix(seq_len(15) * 10, ncol = 3, nrow = 5)
colnames(bulk_counts) <- c("A", "B", "C")
rownames(bulk_counts) <- as.character(seq_len(5))

# Map to bin
result <- map_best_bin(blase_data, "B", bulk_counts)
result

# Map all bulks to bin
results <- map_all_best_bins(blase_data, bulk_counts)

# Plot Heatmap
plot_mapping_result_heatmap(list(result))

# Plot Correlation
plot_mapping_result_corr(result)

# Plot populations
sce <- assign_pseudotime_bins(
    sce,
    pseudotime_slot = "pseudotime", n_bins = 4
)
plot_bin_population(sce, best_bin(result), group_by_slot = "cell_type")

# Getters
bulk_name(result)
best_bin(result)
best_correlation(result)
top_2_distance(result)
strong_mapping(result)
mapping_history(result)
bootstrap_iterations(result)
metric(result)

# Setters
bulk_name(result) <- "New Name"
```

`tradeSeq_BLASE_example_sce` *TradeSeq Example SCE for BLASE Vignette*

Description

Data from the TradeSeq vignette, with the following additional processing applied:

Usage

```
tradeSeq_BLASE_example_sce
```

Format

An object of class `SingleCellExperiment` with 240 rows and 1565 columns.

Details

1. Pseudotime calculated
2. TradeSeq applied
3. Log normalised and normalised counts calculated
4. Erythrocyte cell type removed
5. UMAP calculated

Source

https://bioconductor.org/packages/devel/bioc/vignettes/tradeSeq/inst/doc/tradeSeq.html

---

zhang_2021_heat_shock_bulk

Zhang 2021 Plasmodium falciparum heat shock bulk data

Description

Data originally from https://doi.org/10.1038/s41467-021-24814-1. Used as generated in the BLASE reproducibility documents available at https://zenodo.org/records/16615703, however genes have been subset to reduce file size.

Usage

```
zhang_2021_heat_shock_bulk
```

Format

An object of class `data.frame` with 990 rows and 12 columns.

Source

https://zenodo.org/records/16615703

Index

∗ **annotation**
annotate_sce, 3
∗ **blase-object**
as.BlaseData, 4
assign_pseudotime_bins, 5
bins, 11
bins<-, 11
BlaseData-class, 12
genes, 23
genes<-, 23
pseudobulk_bins, 50
pseudobulk_bins<-, 51
show,BlaseData-method, 51
∗ **datasets**
MCA_PF_SCE, 35
painter_microarray, 37
tradeSeq_BLASE_example_sce, 58
zhang_2021_heat_shock_bulk, 59
∗ **data**
MCA_PF_SCE, 35
painter_microarray, 37
tradeSeq_BLASE_example_sce, 58
zhang_2021_heat_shock_bulk, 59
∗ **gene-selection**
calculate_gene_peakedness, 16
gene_peakedness_spread_selection, 24
get_top_n_genes, 27
plot_gene_peakedness, 41
smooth_gene, 53
∗ **internal**
bins, 11
bins<-, 11
PRIVATE_.ci, 47
PRIVATE_correlation.ci, 48
PRIVATE_cosine_similarity.ci, 49
PRIVATE_distance.ci, 49
pseudobulk_bins, 50
pseudobulk_bins<-, 51
∗ **mapping-result-object**
best_bin, 8
best_correlation, 9
bootstrap_iterations, 13
bulk_name, 14
bulk_name<-, 16
mapping_history, 30
MappingResult, 28
metric, 36
show,MappingResult-method, 52
strong_mapping, 55
top_2_distance, 57
∗ **mapping_plots**
plot_bin_population, 38
plot_mapping_result, 43
plot_mapping_result_corr, 44
plot_mapping_result_heatmap, 45
∗ **mapping**
map_all_best_bins, 31
map_best_bin, 33
∗ **tuning**
evaluate_parameters, 18
evaluate_top_n_genes, 20
find_best_params, 21
plot_find_best_params_results, 39
∗ **util**
get_bins_as_bulk, 26

annotate_sce, 3
as.BlaseData, 4
as.BlaseData(), *12*, *21*
as.BlaseData,SingleCellExperiment-method *(as.BlaseData)*, 4
assign_pseudotime_bins, 5
assign_pseudotime_bins(), *5*
assign_pseudotime_bins,data.frame-method *(assign_pseudotime_bins)*, 5
assign_pseudotime_bins,Seurat-method *(assign_pseudotime_bins)*, 5

assign_pseudotime_bins,SingleCellExperiment-method (assign_pseudotime_bins), 5

best_bin, 8
best_bin,MappingResult-method (best_bin), 8
best_correlation, 9
best_correlation,MappingResult-method (best_correlation), 9
bins, 11
bins,BlaseData-method (bins), 11
bins<-, 11
bins<-,BlaseData-method (bins<-), 11
BiocParallel::BiocParallelParam, *17*, *19*, *21*, *32*
BiocParallel::SerialParam, *17*, *19*, *21*
BiocParallel::SerialParam(), *32*
BlaseData, *5*, *11*, *12*, *19*, *20*, *23*, *24*, *32*, *34*, *50–52*
BlaseData (BlaseData-class), 12
BlaseData-class, 12
bootstrap_iterations, 13
bootstrap_iterations,MappingResult-method (bootstrap_iterations), 13
bulk_name, 14
bulk_name,MappingResult-method (bulk_name), 14
bulk_name<-, 16
bulk_name<-,MappingResult-method (bulk_name<-), 16

calculate_gene_peakedness, 16
calculate_gene_peakedness(), *24*

data.frame, *12*

evaluate_parameters, 18
evaluate_top_n_genes, 20

find_best_params, 21
find_best_params(), *40*

gene_peakedness_spread_selection, 24
genes, 23
genes,BlaseData-method (genes), 23
genes<-, 23
genes<-,BlaseData-method (genes<-), 23
get_bins_as_bulk, 26
get_top_n_genes, 27
ggplot2::ggplot2, *20*, *41*, *44*, *46*

map_all_best_bins, 31
map_best_bin, 33
map_best_bin(), *28*, *29*, *32*
mapping_history, 30
mapping_history,MappingResult-method (mapping_history), 30
MappingResult, *3*, *8*, *9*, *13*, *14*, *16*, 28, *30*, *32*, *34*, *36*, *43–46*, *52*, *56*, *57*
MappingResult-class (MappingResult), 28
MCA_PF_SCE, 35
metric, 36
metric,MappingResult-method (metric), 36

painter_microarray, 37
plot_bin_population, 38
plot_bin_population(), *43*
plot_bin_population,SingleCellExperiment-method (plot_bin_population), 38
plot_find_best_params_results, 39
plot_find_best_params_results(), *22*
plot_gene_peakedness, 41
plot_mapping_result, 42
plot_mapping_result,SingleCellExperiment,MappingResult-met (plot_mapping_result), 43
plot_mapping_result_corr, 44
plot_mapping_result_corr(), *43*
plot_mapping_result_heatmap, 45
PRIVATE_.ci, 47
PRIVATE_correlation.ci, 48
PRIVATE_cosine_similarity.ci, 49
PRIVATE_distance.ci, 49
pseudobulk_bins, 50
pseudobulk_bins,BlaseData-method (pseudobulk_bins), 50
pseudobulk_bins<-, 51
pseudobulk_bins<-,BlaseData-method (pseudobulk_bins<-), 51

scater::plotUMAP(), *43*
show,BlaseData-method, 51
show,MappingResult-method, 52
SingleCellExperiment::SingleCellExperiment, *3–6*, *17*, *24*, *26*, *38*, *41*, *43*, *54*
smooth_gene, 53
strong_mapping, 55
strong_mapping,MappingResult-method (strong_mapping), 55

top_2_distance, 57

top_2_distance,MappingResult-method
(top_2_distance), 57
tradeSeq::associationTest, *27*
tradeSeq_BLASE_example_sce, 58

zhang_2021_heat_shock_bulk, 59