

# Package ‘blase’

June 15, 2026

**Title** Bulk Linking Analysis for Single-cell Experiments

**Version** 1.3.0

**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, ami, 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** devel

**git\_last\_commit** 55a7e55

**git\_last\_commit\_date** 2026-04-28

**Repository** Bioconductor 3.24

**Date/Publication** 2026-06-15

**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 . . . . .	7
best_correlation . . . . .	8
bins . . . . .	10
bins<- . . . . .	10
BlaseData-class . . . . .	11
bootstrap_iterations . . . . .	11
bulk_name . . . . .	13
bulk_name<- . . . . .	14
calculate_gene_peakedness . . . . .	15
evaluate_parameters . . . . .	17
evaluate_top_n_genes . . . . .	18
find_best_params . . . . .	19
genes . . . . .	21
genes<- . . . . .	21
gene_peakedness_spread_selection . . . . .	22
get_bins_as_bulk . . . . .	24
get_top_n_genes . . . . .	25
MappingResult . . . . .	26
mapping_history . . . . .	27
map_all_best_bins . . . . .	29
map_best_bin . . . . .	31
MCA_PF_SCE . . . . .	32
metric . . . . .	33
painter_microarray . . . . .	34
plot_bin_population . . . . .	35
plot_find_best_params_results . . . . .	36
plot_gene_peakedness . . . . .	37
plot_mapping_result . . . . .	39
plot_mapping_result_corr . . . . .	40
plot_mapping_result_heatmap . . . . .	42
PRIVATE_ci . . . . .	43
PRIVATE_correlation.ci . . . . .	44
PRIVATE_cosine_similarity.ci . . . . .	45
PRIVATE_distance.ci . . . . .	45
pseudobulk_bins . . . . .	46
pseudobulk_bins<- . . . . .	47

<code>annotate_sce</code>	3
<code>show,BlaseData-method</code>	47
<code>show,MappingResult-method</code>	48
<code>smooth_gene</code>	49
<code>strong_mapping</code>	51
<code>top_2_distance</code>	52
<code>tradeSeq_BLASE_example_sce</code>	54
<code>zhang_2021_heat_shock_bulk</code>	54
<b>Index</b>	<b>55</b>

---

<code>annotate_sce</code>	<i>Annotate a SCE with BLASE Mappings</i>
---------------------------	---

---

## 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

<code>sce</code>	The <a href="#">SingleCellExperiment::SingleCellExperiment</a> to annotate.
<code>blase_results</code>	A list of <a href="#">MappingResult</a> to use for the annotation.
<code>annotation_col</code>	String. The name of the metadata column in which to store the new annotations.
<code>include_stats</code>	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 <code>SingleCellExperiment::SingleCellExperiment</code> slot(s) containing pseudotime values for each cell to be passed to <code>assign_pseudotime_bins()</code> .
n_bins	Integer. The number of bins to create, passed to <code>assign_pseudotime_bins()</code> .
split_by	String. The <code>split_by</code> method to be passed on to <code>assign_pseudotime_bins()</code> . Must be one of <code>pseudotime_range</code> or <code>cells</code> .

**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

<code>x</code>	An object to add metadata to.
<code>split_by</code>	String. The technique used to split the bins. The default <code>pseudotime_range</code> picks the bin for a cell based on a constant range of pseudotime. <code>cells</code> picks the bin for a cell based on an even number of cells per bin.
<code>n_bins</code>	Integer. The number of bins to split the cells into.
<code>pseudotime_slot</code>	String or Vector of Strings. The name of the <a href="#">SingleCellExperiment::SingleCellExperiment</a> slot(s) containing the pseudotime values for each cell.
<code>...</code>	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	<i>Get name of bulk of a BLASE Mapping Results object.</i>
-----------	--

---

## 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	<a href="#">SingleCellExperiment::SingleCellExperiment</a> 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 <a href="#">BiocParallel::BiocParallelParam</a> for parallelisation. Defaults to <a href="#">BiocParallel::SerialParam</a> .

**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

<code>blase_data</code>	The <a href="#">BlaseData</a> object to use.
<code>bootstrap_iterations</code>	Integer. Iterations for bootstrapping when calculating strong mappings.
<code>BPPARAM</code>	The <a href="#">BiocParallel::BiocParallelParam</a> configuration. Defaults to <a href="#">BiocParallel::SerialParam</a>
<code>make_plot</code>	Boolean. Whether or not to render the plot showing the correlations for each pseudobulk bin when we try to map the given bin.
<code>plot_columns</code>	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 <a href="#">BlaseData</a> 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 <a href="#">BiocParallel::BiocParallelParam</a> . Defaults to <a href="#">BiocParallel::SerialParam</a>
...	params to be passed to child functions, see <a href="#">as.BlaseData()</a>

### 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 metadata 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 <a href="#">tradeSeq::associationTest</a> .
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

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

```

**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 <a href="#">BlaseData</a> 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 <a href="#">BiocParallel::BiocParallelParam</a> for multithreading if desired. Defaults to <a href="#">BiocParallel::SerialParam()</a>
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 <a href="#">BlaseData</a> 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	<i>Painter 2018 Plasmodium falciparum 48h asexual lifecycle microarray data</i>
--------------------	---

---

## 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 <a href="#">SingleCellExperiment::SingleCellExperiment</a> 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 horizontal 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 <a href="#">MappingResult</a> object to plot
...	additional arguments passed to object-specific methods.
group_by_slot	String. The metadata column in the <a href="#">SingleCellExperiment::SingleCellExperiment</a> to be used as the coloring for the output plot. Passed to <a href="#">scater::plotUMAP()</a> as <code>colour_by</code> , 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 <a href="#">MappingResult</a> objects to include in the heatmap.
heatmap_fill_scale	The ggplot2 compatible fill gradient scale to apply to the heatmap.
annotate_strong	Boolean. 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 bootstrapping. 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                numeric vector (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 bootstrapping. 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	numeric vector (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 bootstrapping. 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	numeric vector (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 <a href="#">BlaseData</a> 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 <a href="#">BlaseData</a> 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	<a href="#">SingleCellExperiment::SingleCellExperiment</a> to do the calculations on.
gene	String. The name of the gene to smooth
pseudotime_slot	String. The slot in the <a href="#">SingleCellExperiment::SingleCellExperiment</a> object meta-data 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	<i>Get the difference in correlation between the top 2 most correlated bins for a BLASE Mapping Results object.</i>
----------------	---

---

## 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, 10
  - bins<-, 10
  - BlaseData-class, 11
  - genes, 21
  - genes<-, 21
  - pseudobulk\_bins, 46
  - pseudobulk\_bins<-, 47
  - show, BlaseData-method, 47
- \* **datasets**
  - MCA\_PF\_SCE, 32
  - painter\_microarray, 34
  - tradeSeq\_BLASE\_example\_sce, 54
  - zhang\_2021\_heat\_shock\_bulk, 54
- \* **data**
  - MCA\_PF\_SCE, 32
  - painter\_microarray, 34
  - tradeSeq\_BLASE\_example\_sce, 54
  - zhang\_2021\_heat\_shock\_bulk, 54
- \* **gene-selection**
  - calculate\_gene\_peakedness, 15
  - gene\_peakedness\_spread\_selection, 22
  - get\_top\_n\_genes, 25
  - plot\_gene\_peakedness, 37
  - smooth\_gene, 49
- \* **internal**
  - bins, 10
  - bins<-, 10
  - PRIVATE\_.ci, 43
  - PRIVATE\_correlation.ci, 44
  - PRIVATE\_cosine\_similarity.ci, 45
  - PRIVATE\_distance.ci, 45
  - pseudobulk\_bins, 46
  - pseudobulk\_bins<-, 47
- \* **mapping-result-object**
  - best\_bin, 7
  - best\_correlation, 8
  - bootstrap\_iterations, 11
  - bulk\_name, 13
  - bulk\_name<-, 14
  - mapping\_history, 27
  - MappingResult, 26
  - metric, 33
  - show, MappingResult-method, 48
  - strong\_mapping, 51
  - top\_2\_distance, 52
- \* **mapping\_plots**
  - plot\_bin\_population, 35
  - plot\_mapping\_result, 39
  - plot\_mapping\_result\_corr, 40
  - plot\_mapping\_result\_heatmap, 42
- \* **mapping**
  - map\_all\_best\_bins, 29
  - map\_best\_bin, 31
- \* **tuning**
  - evaluate\_parameters, 17
  - evaluate\_top\_n\_genes, 18
  - find\_best\_params, 19
  - plot\_find\_best\_params\_results, 36
- \* **util**
  - get\_bins\_as\_bulk, 24
- annotate\_sce, 3
- as.BlaseData, 4
- as.BlaseData(), 11, 20
- as.BlaseData, SingleCellExperiment-method (as.BlaseData), 4
- assign\_pseudotime\_bins, 5
- assign\_pseudotime\_bins(), 4
- 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, 7
- best\_bin, MappingResult-method (best\_bin), 7
- best\_correlation, 8
- best\_correlation, MappingResult-method (best\_correlation), 8

- bins, 10
- bins,BlaseData-method (bins), 10
- bins<-, 10
- bins<- ,BlaseData-method (bins<-), 10
- BiocParallel::BiocParallelParam, 15, 17, 20, 29
- BiocParallel::SerialParam, 15, 17, 20
- BiocParallel::SerialParam(), 29
- BlaseData, 5, 10, 11, 17, 18, 21, 29, 31, 46, 47
- BlaseData (BlaseData-class), 11
- BlaseData-class, 11
- bootstrap\_iterations, 11
- bootstrap\_iterations,MappingResult-method (bootstrap\_iterations), 11
- bulk\_name, 13
- bulk\_name,MappingResult-method (bulk\_name), 13
- bulk\_name<-, 14
- bulk\_name<- ,MappingResult-method (bulk\_name<-), 14
- calculate\_gene\_peakedness, 15
- calculate\_gene\_peakedness(), 22
- data.frame, 11
- evaluate\_parameters, 17
- evaluate\_top\_n\_genes, 18
- find\_best\_params, 19
- find\_best\_params(), 36, 37
- gene\_peakedness\_spread\_selection, 22
- genes, 21
- genes,BlaseData-method (genes), 21
- genes<-, 21
- genes<- ,BlaseData-method (genes<-), 21
- get\_bins\_as\_bulk, 24
- get\_top\_n\_genes, 25
- ggplot2::ggplot2, 18, 38, 41, 42
- map\_all\_best\_bins, 29
- map\_best\_bin, 31
- map\_best\_bin(), 26, 30
- mapping\_history, 27
- mapping\_history,MappingResult-method (mapping\_history), 27
- MappingResult, 3, 7, 9, 12–14, 26, 28, 29, 31, 33, 39, 40, 42, 48, 51, 52
- MappingResult-class (MappingResult), 26
- MCA\_PF\_SCE, 32
- metric, 33
- metric,MappingResult-method (metric), 33
- painter\_microarray, 34
- plot\_bin\_population, 35
- plot\_bin\_population(), 40
- plot\_bin\_population,SingleCellExperiment-method (plot\_bin\_population), 35
- plot\_find\_best\_params\_results, 36
- plot\_find\_best\_params\_results(), 20
- plot\_gene\_peakedness, 37
- plot\_mapping\_result, 39
- plot\_mapping\_result,SingleCellExperiment,MappingResult-method (plot\_mapping\_result), 39
- plot\_mapping\_result\_corr, 40
- plot\_mapping\_result\_corr(), 40
- plot\_mapping\_result\_heatmap, 42
- PRIVATE.ci, 43
- PRIVATE\_correlation.ci, 44
- PRIVATE\_cosine\_similarity.ci, 45
- PRIVATE\_distance.ci, 45
- pseudobulk\_bins, 46
- pseudobulk\_bins,BlaseData-method (pseudobulk\_bins), 46
- pseudobulk\_bins<-, 47
- pseudobulk\_bins<- ,BlaseData-method (pseudobulk\_bins<-), 47
- scater::plotUMAP(), 39
- show,BlaseData-method, 47
- show,MappingResult-method, 48
- SingleCellExperiment::SingleCellExperiment, 3, 4, 6, 15, 22, 24, 35, 38, 39, 49
- smooth\_gene, 49
- strong\_mapping, 51
- strong\_mapping,MappingResult-method (strong\_mapping), 51
- top\_2\_distance, 52
- top\_2\_distance,MappingResult-method (top\_2\_distance), 52
- tradeSeq::associationTest, 25
- tradeSeq\_BLASE\_example\_sce, 54
- zhang\_2021\_heat\_shock\_bulk, 54