

# Package ‘scBubbletree’

May 28, 2025

**Type** Package

**Title** Quantitative visual exploration of scRNA-seq data

**Version** 1.11.0

**Description** scBubbletree is a quantitative method for the visual exploration of scRNA-seq data, preserving key biological properties such as local and global cell distances and cell density distributions across samples. It effectively resolves overplotting and enables the visualization of diverse cell attributes from multiomic single-cell experiments. Additionally, scBubbletree is user-friendly and integrates seamlessly with popular scRNA-seq analysis tools, facilitating comprehensive and intuitive data interpretation.

**License** GPL-3 + file LICENSE

**Depends** R (>= 4.2.0)

**Imports** reshape2, BiocParallel, ape, scales, Seurat, ggplot2, ggtree, patchwork, proxy, methods, stats, base, utils, dplyr

**Suggests** BiocStyle, knitr, testthat, cluster, SingleCellExperiment

**Encoding** UTF-8

**NeedsCompilation** no

**biocViews** Visualization,Clustering, SingleCell,Transcriptomics,RNASeq

**BugReports** <https://github.com/snaketron/scBubbletree/issues>

**URL** <https://github.com/snaketron/scBubbletree>

**SystemRequirements** Python (>= 3.6), leidenalg (>= 0.8.2)

**RoxygenNote** 6.1.1

**VignetteBuilder** knitr

**git\_url** <https://git.bioconductor.org/packages/scBubbletree>

**git\_branch** devel

**git\_last\_commit** b55a5b7

**git\_last\_commit\_date** 2025-04-15

**Repository** Bioconductor 3.22

**Date/Publication** 2025-05-27

**Author** Simo Kitanovski [aut, cre]

**Maintainer** Simo Kitanovski <simokitanovski@gmail.com>

## Contents

scBubbletree-package . . . . .	2
compare_bubbletrees . . . . .	3
d_500 . . . . .	4
d_ccl . . . . .	5
get_bubbletree_dummy . . . . .	6
get_bubbletree_graph . . . . .	8
get_bubbletree_kmeans . . . . .	11
get_cat_tiles . . . . .	13
get_gini . . . . .	15
get_gini_k . . . . .	16
get_k . . . . .	18
get_num_cell_tiles . . . . .	19
get_num_tiles . . . . .	21
get_num_violins . . . . .	22
get_r . . . . .	24
<b>Index</b>	<b>26</b>

---

scBubbletree-package    *The R package scBubbletree*

---

## Description

Method for quantitative visualization of single cell RNA-seq data

## Details

This package contains functions for clustering, hierarchical grouping of clusters and visualization of scRNA-seq data.

## Author(s)

Authors and maintainers:

- Simo Kitanovski <simokitanovski@uni-due.de> ([ORCID](#))

## See Also

Useful links:

- <https://github.com/snaketron/scBubbletree>
- Report bugs at <https://github.com/snaketron/scBubbletree/issues>

---

compare_bubbletrees	<i>Comparison of two bubbletrees generated from the same scRNA-seq data</i>
---------------------	---

---

## Description

compare\_bubbletrees takes as its main input two bubbletrees generated from the **same input data** but potentially with different input parameters (e.g. clustering method or resolutions).

It then does the following two operations:

1. computes the Jaccard distance (JD) and the intersection between paris of clusters from the two bubbletrees. This is visualized as a heatmap.
2. it visualizes the two bubbletrees together with the heatmap.

## Usage

```
compare_bubbletrees(btd_1,
                    btd_2,
                    tile_bw = FALSE,
                    tile_text_size = 3,
                    ratio_heatmap = 0.5)
```

## Arguments

btd_1	bubbletree object
btd_2	bubbletree object
tile_text_size	integer, size of tile labels (default = 3)
tile_bw	logical, tile grayscale (tile_bw = TRUE) vs. color (tile_bw = FALSE, default)
ratio_heatmap	nummeric, probability (default = 0.5) that dictates the relative width and height of the heatmap and the bubbletrees

## Details

compare\_bubbletrees takes as its main input two bubbletrees generated from the **same input data** but potentially with different input parameters (e.g. clustering method or resolutions).

It then does the following two operations:

1. computes the Jaccard distance and the intersection between paris of clusters from the two bubbletrees. This is visualized as a heatmap.
2. it visualizes the two bubbletrees together with the heatmap.

## Value

comparison	ggplot2 objects assembled by R-package patchwork
m	data.frame object with JD and intersection for each pair of clusters from the two bubbletrees

## Author(s)

Simo Kitanovski <simo.kitanovski@uni-due.de>

**See Also**

get\_k, get\_bubbletree\_dummy, get\_bubbletree\_graph, get\_bubbletree\_kmeans, get\_gini, get\_gini\_k, d\_500, get\_num\_tiles, get\_num\_violins, get\_cat\_tiles

**Examples**

```
# input data
data("d_500", package = "scBubbletree")
A <- d_500$A

btd_1 <- get_bubbletree_graph(x = A,
                             r = 1,
                             n_start = 20,
                             iter_max = 100,
                             algorithm = "original",
                             knn_k = 50,
                             hclust_method = "average",
                             hclust_distance = "euclidean",
                             cores = 1,
                             round_digits = 2,
                             show_simple_count = FALSE)

btd_2 <- get_bubbletree_kmeans(x = A,
                              k = 8,
                              cores = 1,
                              round_digits = 1,
                              show_simple_count = FALSE,
                              kmeans_algorithm = "MacQueen",
                              hclust_distance = "euclidean",
                              hclust_method = "average")

btd_comparison <- compare_bubbletrees(btd_1 = btd_1,
                                       btd_2 = btd_2,
                                       tile_bw = FALSE,
                                       tile_text_size = 3,
                                       ratio_heatmap = 0.5)

# plot
btd_comparison$tree_comparison

# data.frame of heatmap data
btd_comparison$m
```

---

d\_500

*Dataset: 500 PBMCs*


---

**Description**

d\_500 is a list with 3 elements:

1. A = numeric matrix  $A^{500 \times 15}$  with n=500 rows for PBMCs and f=15 principal components.

2.  $f$  = character vector  $f$  of length 500. Each element in  $f$  represents the predicted cell type of a specific cell.
3.  $fs$  = numeric matrix containing normalized gene expressions of 12 marker genes in 500 cells.

### Usage

```
data("d_500", package = "scBubbletree")
```

### Format

Format of d\_500: list

### Details

This data is a sample drawn from a larger dataset of 2,700 PBMCs. The original dataset was processed as described in vignette (accessed 23, Sep, 2022):

[https://satijalab.org/seurat/articles/multimodal\\_reference\\_mapping.html](https://satijalab.org/seurat/articles/multimodal_reference_mapping.html)

See R script inst/script/get\_d\_500.R to see how this dataset was created.

### Source

[https://satijalab.org/seurat/articles/multimodal\\_reference\\_mapping.html](https://satijalab.org/seurat/articles/multimodal_reference_mapping.html)

### Examples

```
data("d_500", package = "scBubbletree")
```

```
A <- d_500$A
base::dim(A)
```

```
f <- d_500$f
base::table(f)
```

```
fs <- d_500$fs
base::dim(fs)
```

---

d\_ccl

*Dataset: scRNA-seq data of 3,918 cells from 5 adenocarcinoma cell lines*

---

### Description

d\_ccl is a list with 3 elements:

1. A = numeric matrix with n=3,918 rows for cells and f=15 principal components
2. m = data.frame meta data
3. e = numeric matrix containing normalized gene expressions of 5 marker genes

### Usage

```
data("d_ccl", package = "scBubbletree")
```

## Format

Format of d\_ccl: list

## Details

d\_ccl is a scRNA-seq dataset containing a mixture of 3,918 cells from five human lung adenocarcinoma cell lines (HCC827, H1975, A549, H838 and H2228). The dataset is available here:

[https://github.com/LuyiTian/sc\\_mixology/blob/master/data/sincell\\_with\\_class\\_5cl.RData](https://github.com/LuyiTian/sc_mixology/blob/master/data/sincell_with_class_5cl.RData)

The library has been prepared with 10x Chromium platform and sequenced with Illumina NextSeq 500 platform. Raw data has been processed with Cellranger. The tool demuxlet has been used to predict the identity of each cell based on known genetic differences between the different cell lines.

See R script inst/script/get\_d\_ccl.R to see how this dataset was created.

## Source

[https://github.com/LuyiTian/sc\\_mixology/blob/master/data/sincell\\_with\\_class\\_5cl.RData](https://github.com/LuyiTian/sc_mixology/blob/master/data/sincell_with_class_5cl.RData)

## References

Tian, Luyi, et al. "Benchmarking single cell RNA-sequencing analysis pipelines using mixture control experiments." Nature methods 16.6 (2019): 479-487

## Examples

```
data("d_ccl", package = "scBubbletree")
```

```
A <- d_ccl$A
base::dim(A)
```

```
m <- d_ccl$m
utils::head(m)
```

```
e <- d_ccl$e
base::dim(e)
```

---

get_bubbletree_dummy	<i>Build bubbletree from matrix A of low-dimensional projections and vector cs of externally generated cluster IDs</i>
----------------------	--

---

## Description

get\_bubbletree\_dummy takes two main inputs:

1. numeric matrix  $A^{n \times f}$ , which represents a low-dimensional projection (obtained e.g. by PCA) of the original high-dimensional scRNA-seq data, with  $n$  rows as cells and  $f$  columns as low-dimension features.
2. vector  $cs$  of cluster IDs of each cell

The function get\_bubbletree\_dummy performs one main operation. It organizes the bubbles (defined by  $cs$ ) in a hierarchical dendrogram (bubbletree) which represents the hierarchical relationships between the clusters (bubbles).

**Usage**

```
get_bubbletree_dummy(x,
                     cs,
                     B = 200,
                     N_eff = 100,
                     hclust_distance = "euclidean",
                     hclust_method = "average",
                     cores = 1,
                     round_digits = 2,
                     show_simple_count = FALSE,
                     verbose = TRUE)
```

**Arguments**

x	numeric matrix ( $A^{n \times f}$ with $n$ cells, and $f$ low-dimensional projections of the original single cell RNA-seq dataset)
cs	vector, cluster IDs
B	integer, number of bootstrap iterations to perform in order to generate bubbletree
N_eff	integer, number of cells to draw randomly from each cluster when computing inter-cluster distances
hclust_distance	distance measure to be used: euclidean (default) or manhattan, see documentation of <code>stats::dist</code>
hclust_method	agglomeration method to be used, default = average. See documentation of <code>stats::hclust</code>
cores	integer, number of PC cores for parallel execution
round_digits	integer, number of decimal places to keep when showing the relative frequency of cells in each bubble
show_simple_count	logical, if <code>show_simple_count=T</code> , cell counts in each bubble will be divided by 1,000 to improve readability. This is only useful for samples that are composed of millions of cells.
verbose	logical, progress messages

**Details**

This function is similar to `get_bubbletree_kmeans` and `get_bubbletree_graph` but skips the clustering step. See the documentation of the respective functions.

**Value**

A	input x matrix
k	number of clusters
km	NULL
ph	boot_ph: bootstrap dendrograms $H_b$ ; main_ph: bubbletree $H$
ph_data	two phlogenies: ph_c = phylogeny constructed from bubble centroids (computed from $A^{n \times f}$ ); ph_p = main_ph = phylogeny constructed from intercell distances
pair_dist	inter-cluster distances used to generate the dendrograms

cluster	cluster assignments of each cell
input_par	list of all input parameters
tree	ggtree bubbletree object
tree_simple	simplified ggtree bubbletree object
tree_meta	meta-data associated with the bubbletree

**Author(s)**

Simo Kitanovski <simon.kitanovski@uni-due.de>

**See Also**

get\_k, get\_r, get\_bubbletree\_kmeans, get\_bubbletree\_graph, get\_bubbletree\_comparison, get\_gini, get\_gini\_k, get\_num\_tiles, get\_num\_violins, get\_cat\_tiles, d\_500

**Examples**

```
# input data
data("d_500", package = "scBubbletree")
A <- d_500$A

cs <- base::sample(x = LETTERS[1:5], size = nrow(A), replace = TRUE)

db <- get_bubbletree_dummy(x = A,
                           cs = cs,
                           B = 200,
                           N_eff = 100,
                           hclust_distance = "euclidean",
                           hclust_method = "average",
                           cores = 1)
```

---

get\_bubbletree\_graph    *Louvain clustering and hierarchical grouping of  $k'$  clusters (bubbles)*

---

**Description**

get\_bubbletree\_graph has two main inputs:

1. numeric matrix  $A^{n \times f}$ , which represents a low-dimensional projection (obtained e.g. by PCA) of the original high-dimensional scRNA-seq data, with  $n$  rows as cells and  $f$  columns as low-dimension features.
2. clustering resolution  $r$

The function get\_bubbletree\_graph performs two main operations. First, it performs Louvain clustering to identify groups (bubbles) of transcriptionally similar cells; second, it organizes the bubbles in a hierarchical dendrogram (bubbletree) which adequately represents inter-cluster relationships.



**Usage**

```
get_bubbletree_graph(x,
                     r,
                     B = 200,
                     N_eff = 100,
                     n_start = 20,
                     iter_max = 100,
                     algorithm = "original",
                     knn_k = 20,
                     hclust_method = "average",
                     hclust_distance = "euclidean",
                     cores = 1,
                     round_digits = 2,
                     show_simple_count = FALSE,
                     verbose = TRUE)
```

**Arguments**

x	numeric matrix ( $A^{n \times f}$ with $n$ cells, and $f$ low-dimensional projections of the original single cell RNA-seq dataset)
r	number, clustering resolution
B	integer, number of bootstrap iterations to perform in order to generate bubble-tree. If $B = 200$ (default), cluster centroids are used to compute inter-cluster distances and $N_{eff}$ is ignored, i.e. all cells are used to compute centroids.
N_eff	integer, number of cells to draw randomly from each cluster when computing inter-cluster distances.
n_start, iter_max	parameters for Louvain clustering, see documentation of function FindClusters, R-package Seurat
algorithm	character, four clustering algorithms: 'original', 'LMR', 'SLM' and 'Leiden', see documentation of function FindClusters, R-package Seurat
knn_k	integer, defines $k$ for the k-nearest neighbor algorithm, see documentation of function FindClusters, R-package Seurat
hclust_method	the agglomeration method to be used (default = average). See documentation of stats::hclust
hclust_distance	distance measure to be used: euclidean (default) or manhattan, see documentation of stats::dist
cores	integer, number of PC cores for parallel execution
round_digits	integer, number of decimal places to keep when showing the relative frequency of cells in each bubble
show_simple_count	logical, if show_simple_count=T, cell counts in each bubble will be divided by 1,000 to improve readability. This is only useful for samples that are composed of millions of cells.
verbose	logical, progress messages

## Details

For Louvain clustering `get_bubbletree_graph` uses the function `FindClusters` implemented in R-package `Seurat`. For additional information on the clustering procedure see the documentation of `FindClusters`. To organize the resulting clusters in a hierarchical dendrogram, then the following steps are performed:

1. In bootstrap iteration  $b$  from  $1 : B$
2. draw up to  $N_{eff}$  number of cells at random from each cluster without replacement
3. compute distances (in space  $A^{n \times f}$ ) between all pairs of cells in cluster  $i$  and cluster  $j$
4. compute mean distance between cluster  $i$  and  $j$  and populate inter-cluster distance matrix  $D_b^{k \times k}$
5. perform hierarchical clustering with user-specified agglomeration method based on  $D_b^{k \times k}$  to generate dendrogram  $H_b$
6. quantify branch robustness in  $H$  by counting how many times each branch is found among bootstrap dendrograms  $H_b$

## Value

<code>A</code>	input x matrix
<code>k</code>	number of clusters
<code>r</code>	clustering resolution
<code>ph</code>	<code>boot_ph</code> : bootstrap dendrograms $H_b$ ; <code>main_ph</code> : bubbletree $H$
<code>ph_data</code>	two phlogenies: <code>ph_c</code> = phylogeny constructed from bubble centroids (computed from $A^{n \times f}$ ); <code>ph_p</code> = <code>main_ph</code> = phylogeny constructed from intercell distances
<code>pair_dist</code>	inter-cluster distances used to generate the dendrograms
<code>cluster</code>	cluster assignments of each cell
<code>input_par</code>	list of all input parameters
<code>tree</code>	ggtree bubbletree object
<code>tree_simple</code>	simplified ggtree bubbletree object
<code>tree_meta</code>	meta-data associated with the bubbletree

## Author(s)

Simo Kitanovski <[simo.kitanovski@uni-due.de](mailto:simo.kitanovski@uni-due.de)>

## See Also

`get_k`, `get_bubbletree_dummy`, `get_bubbletree_kmeans`, `get_bubbletree_comparison`, `get_gini`, `get_gini_k`, `d_500`, `get_num_tiles`, `get_num_violins`, `get_cat_tiles`

## Examples

```
# input data
data("d_500", package = "scBubbletree")
A <- d_500$A

b <- get_bubbletree_graph(x = A,
                          r = 1,
                          B = 200,
```

```

        N_eff = 100,
        n_start = 20,
        iter_max = 100,
        algorithm = "original",
        knn_k = 20,
        hclust_method = "average",
        hclust_distance = "euclidean",
        cores = 1,
        round_digits = 2,
        show_simple_count = FALSE)

b$tree

```

---

get\_bubbletree\_kmeans *k-means clustering and hierarchical grouping of  $k$  clusters (bubbles)*

---

## Description

get\_bubble\_kmeans takes two main inputs:

1. numeric matrix  $A^{n \times f}$ , which represents a low-dimensional projection (obtained e.g. by PCA) of the original high-dimensional scRNA-seq data, with  $n$  rows as cells and  $f$  columns as low-dimension features.
2. number  $k$  of clusters

The function get\_bubble\_kmeans performs two main operations. First, it performs k-means clustering to identify groups (bubbles) of transcriptionally similar cells. Second, it organizes the bubbles in a hierarchical dendrogram (bubbletree) which adequately represents inter-cluster relationships.

## Usage

```

get_bubbletree_kmeans(x,
                      k,
                      B = 200,
                      N_eff = 100,
                      n_start = 1000,
                      iter_max = 300,
                      kmeans_algorithm = "MacQueen",
                      hclust_distance = "euclidean",
                      hclust_method = "average",
                      cores = 1,
                      round_digits = 2,
                      show_simple_count = FALSE,
                      verbose = TRUE)

```

## Arguments

x	numeric matrix ( $A^{n \times f}$ with $n$ cells, and $f$ low-dimensional projections of the original single cell RNA-seq dataset)
k	integer, number of clusters
B	integer, number of bootstrap iterations to perform in order to generate bubbletree
N_eff	integer, number of cells to draw randomly from each cluster when computing inter-cluster distances

n_start, iter_max, kmeans_algorithm	parameters for k-means clustering, see documentation of function k-means, R-package stats
hclust_distance	distance measure to be used: euclidean (default) or manhattan, see documentation of stats::dist
hclust_method	the agglomeration method to be used, default = average. See documentation of stats::hclust
cores	integer, number of PC cores for parallel execution
round_digits	integer, number of decimal places to keep when showing the relative frequency of cells in each bubble
show_simple_count	logical, if show_simple_count=T, cell counts in each bubble will be divided by 1,000 to improve readability. This is only useful for samples that are composed of millions of cells.
verbose	logical, progress messages

### Details

For k-means clustering get\_bubble\_kmeans uses the function kmeans implemented in R-package stats (version 4.2.0). For additional information on the clustering procedure see the documentation of kmeans. To organize the resulting clusters in a hierarchical dendrogram these steps are performed:

1. In bootstrap iteration  $b$  from  $1 : B$
2. draw up to  $N_{eff}$  number of cells at random from each cluster without replacement
3. compute distances (in space  $A^{n \times f}$ ) between pairs of cells in cluster  $i$  and cluster  $j$
4. compute mean distance between cluster  $i$  and  $j$  and populate inter-cluster distance matrix  $D_b^{k \times k}$
5. perform hierarchical clustering with user-specified agglomeration method based on  $D_b^{k \times k}$  to generate dendrogram  $H_b$
6. quantify branch robustness in  $H$  by counting how many times each branch is found among the bootstrap dendrograms  $H_b$

### Value

A	input matrix x
k	number of clusters
km	k-means clustering results identical to those generated by function k-means from R-package stats
ph	boot_ph: bootstrap dendrograms $H_b$ ; main_ph: bubbletree $H$
ph_data	two phlogenies: ph_c = phylogeny constructed from bubble centroids (computed from $A^{n \times f}$ ); ph_p = main_ph = phylogeny constructed from intercell distances
pair_dist	inter-cluster distances used to generate the dendrograms
cluster	cluster assignments of each cell
input_par	list of all input parameters
tree	ggtree bubbletree object
tree_simple	simplified ggtree bubbletree object
tree_meta	meta-data associated with the bubbletree

**Author(s)**

Simo Kitanovski <simo.kitanovski@uni-due.de>

**See Also**

get\_k, get\_bubbletree\_dummy, get\_bubbletree\_graph, get\_gini, get\_gini\_k, d\_500, get\_num\_tiles, get\_num\_violins, get\_cat\_tiles, get\_bubbletree\_comparison

**Examples**

```
# input data
data("d_500", package = "scBubbletree")
A <- d_500$A

b <- get_bubbletree_kmeans(x = A,
                           k = 8,
                           B = 200,
                           N_eff = 100,
                           cores = 1,
                           round_digits = 1,
                           show_simple_count = FALSE,
                           kmeans_algorithm = "MacQueen",
                           hclust_distance = "euclidean",
                           hclust_method = "average")

b$tree
```

---

get\_cat\_tiles

---

*Visualization of categorical cell features using tile plots*


---

**Description**

get\_cat\_tiles creates tile plot to visualize the relative frequency of categorical cell features between and within the bubbles of a bubbletree

**Usage**

```
get_cat_tiles(btd,
              f,
              integrate_vertical,
              round_digits = 2,
              tile_text_size = 3,
              tile_bw = FALSE,
              x_axis_name = "Feature",
              rotate_x_axis_labels = TRUE)
```

**Arguments**

btd	bubbletree object
f	character vector, categorical cell features

integrate_vertical	logical, if integrate_vertical=TRUE: relative frequency of the features is shown in each bubble, if integrate_vertical=FALSE: relative frequencies of the features is shown within each bubble
round_digits	integer, number of decimal places to keep when showing the relative frequency of cells in each bubble
tile_text_size	integer, size of tile labels
x_axis_name	character, x-axis title
rotate_x_axis_labels	logical, should the x-axis labels be shown horizontally (rotate_x_axis_labels = FALSE) or vertically (rotate_x_axis_labels = TRUE)
tile_bw	logical, tile grayscale (tile_bw = TRUE) vs. color (tile_bw = FALSE, default)

### Details

get\_cat\_tiles uses two main inputs:

1. bubbletree object
2. character vector of categorical cell features.

The order of the cells used to generate the bubbletree (input 1.) should correspond to the order of cells in the vector of categorical cell features (input 2.)

This function computes:

1. with integrate\_vertical=T: relative frequencies of each feature across the different bubbles
2. with integrate\_vertical=F: within-bubble relative frequencies (composition) of different features

### Value

plot	ggplot2, tile plot
table	data.frame, raw data used to generate the plot

### Author(s)

Simo Kitanovski <simo.kitanovski@uni-due.de>

### See Also

get\_k, get\_r, get\_bubbletree\_dummy, get\_bubbletree\_kmeans, get\_bubbletree\_graph, get\_gini, get\_gini\_k, get\_num\_tile, get\_num\_violins, d\_500

### Examples

```
# input data
data("d_500", package = "scBubbletree")
A <- d_500$A
f <- d_500$f

b <- get_bubbletree_graph(x = A,
  r = 0.8,
  N_eff = 100)
```

```

g_v <- get_cat_tiles(btd = b,
                    f = f,
                    integrate_vertical = TRUE,
                    round_digits = 2,
                    tile_text_size = 3,
                    x_axis_name = "Feature",
                    rotate_x_axis_labels = TRUE)

g_h <- get_cat_tiles(btd = b,
                    f = f,
                    integrate_vertical = FALSE,
                    round_digits = 2,
                    tile_text_size = 3,
                    x_axis_name = "Feature",
                    rotate_x_axis_labels = TRUE)

b$tree|g_v$plot|g_h$plot

```

get\_gini

*Gini impurity index computed for a clustering solution and a vector of categorical cell feature labels*

## Description

How well is a set of categorical feature labels (e.g. cell type predictions) partitioned accross the different clusters of a clustering solution? We can assess this using the Gini impurity index (see details below).

Inputs are two equal-sized vectors:

- 1) clusters IDs
- 2) labels

Output:

- 1) cluster-specific purity -> Gini impurity (GI) index
- 2) clustering solution impurity -> Weighted Gini impurity (WGI) index

## Usage

```
get_gini(labels, clusters)
```

## Arguments

labels	character or numeric vector of labels
clusters	character or numeric vector of cluster IDs

## Details

To quantify the purity of a cluster (or bubble)  $i$  with  $n_i$  number of cells, each of which carries one of  $L$  possible labels (e.g. cell type), we computed the Gini impurity index:

$$GI_i = \sum_{j=1}^L \pi_{ij}(1 - \pi_{ij}),$$

with  $\pi_{ij}$  as the relative frequency of label  $j$  in cluster  $i$ . In homogeneous ('pure') clusters most cells carry a distinct label. Hence, the  $\pi$ 's are close to either 1 or 0, and GI takes on a small value close to zero. In 'impure' clusters cells carry a mixture of different labels. In this case most  $\pi$  are far from either 1 or 0, and GI diverges from 0 and approaches 1. If the relative frequencies of the different labels in cluster  $i$  are equal to the (background) relative frequencies of the labels in the sample, then cluster  $i$  is completely 'impure'.

To compute the overall Gini impurity of a bubbletree, which represents a clustering solution with  $k$  bubbles, we estimated the weighted Gini impurity (WGI) by computing the weighted (by the cluster size) average of the GIs:

$$WGI = \sum_{i=1}^k GI_i n_i / n,$$

with  $n_i$  as the number of cells in cluster  $i$  and  $n = \sum_i n_i$ .

## Value

gi	Gini impurity of each bubble
wgi	Weighted Gini impurity index of the bubbletree

## Author(s)

Simo Kitanovski <simo.kitanovski@uni-due.de>

## See Also

get\_k, get\_r, get\_bubbletree\_kmeans, get\_bubbletree\_dummy, get\_bubbletree\_graph, get\_gini\_k, d\_500

## Examples

```
get_gini(labels = base::sample(x = LETTERS[1:4], size = 100, replace = TRUE),
         clusters = base::sample(x = letters[1:4], size = 100, replace = TRUE))
```

---

get_gini_k	<i>Gini impurity index computed for a list of clustering solutions obtained by functions get_k or get_r and a vector of categorical cell feature labels</i>
------------	---

---

## Description

Given The Gini impurity (GI) index allows us to quantitatively evaluate how well a set of labels (categorical features) are split across a set of bubbles. We have a completely perfect split (GI = 0) when each bubble is 'pure', i.e. each bubble contains labels coming from distinct a class. In contrast to this, we have completely imperfect split (GI = 1) when the relative frequency distribution of the labels in each bubble is identical to the background relative frequency distribution of the labels.

Cell type predictions are a type of categorical features that are often used to evaluate the goodness of the clustering. get\_gini\_k takes as input: 1) a vector of labels for each cell (e.g. cell types) and



2) object returned by function `get_k` or `get_r`. Then it computes for each  $k$  or  $r$  the cluster purity and weighted gini impurity of each clustering solution mean GI, which is another way of finding an optimal clustering resolution.

### Usage

```
get_gini_k(labels, obj)
```

### Arguments

<code>labels</code>	character/factor vector of labels
<code>obj</code>	object returned by functions <code>get_k</code> or <code>get_r</code>

### Details

To quantify the purity of a cluster (or bubble)  $i$  with  $n_i$  number of cells, each of which carries one of  $L$  possible labels (e.g. cell type), we computed the Gini impurity index:

$$GI_i = \sum_{j=1}^L \pi_{ij}(1 - \pi_{ij}),$$

with  $\pi_{ij}$  as the relative frequency of label  $j$  in cluster  $i$ . In homogeneous ('pure') clusters most cells carry a distinct label. Hence, the  $\pi$ 's are close to either 1 or 0, and GI takes on a small value close to zero. In 'impure' clusters cells carry a mixture of different labels. In this case most  $\pi$  are far from either 1 or 0, and GI diverges from 0 and approaches 1. If the relative frequencies of the different labels in cluster  $i$  are equal to the (background) relative frequencies of the labels in the sample, then cluster  $i$  is completely 'impure'.

To compute the overall Gini impurity of a bubbletree, which represents a clustering solution with  $k$  bubbles, we estimated the weighted Gini impurity (WGI) by computing the weighted (by the cluster size) average of the GIs:

$$WGI = \sum_{i=1}^k GI_i n_i / n,$$

with  $n_i$  as the number of cells in cluster  $i$  and  $n = \sum_i n_i$ .

### Value

<code>gi_summary</code>	GI for each bubble of a clustering solution with clustering resolution $k$ or $r$
<code>wgi_summary</code>	WGI for each clustering solution with clustering resolution $k$ or $r$

### Author(s)

Simo Kitanovski <simo.kitanovski@uni-due.de>

### See Also

`get_k`, `get_r`, `get_gini`, `get_bubbletree_kmeans`, `get_bubbletree_graph`, `get_bubbletree_dummy`, `d_500`, `get_num_tiles`, `get_num_violins`, `get_cat_tiles`

### Examples

```
# input data
data("d_500", package = "scBubbletree")
A <- d_500$A
f <- d_500$f
```

```

b_k <- get_k(x = A,
             ks = 1:5,
             B_gap = 5,
             n_start = 100,
             iter_max = 200,
             kmeans_algorithm = "MacQueen",
             cores = 1)

b_r <- get_r(x = A,
             rs = c(0.1, 0.5, 1),
             B_gap = 5,
             n_start = 20,
             iter_max = 100,
             algorithm = "original",
             cores = 1)

get_gini_k(labels = f, obj = b_k)
get_gini_k(labels = f, obj = b_r)

```

---

get\_k

*Finding optimal number  $k$  of clusters*


---

## Description

To perform k-means clustering we must specify a number  $k$  of clusters. Data-driven metrics, such as the Gap statistic or the within-cluster sum of squares (WCSS), can be used to infer appropriate  $k$  from the data. `get_k` computes the Gap statistic and WCSS for a number of clusters  $ks$ .

## Usage

```

get_k(x,
      ks,
      B_gap = 20,
      n_start = 1000,
      iter_max = 300,
      kmeans_algorithm = "MacQueen",
      cores = 1,
      verbose = TRUE)

```

## Arguments

<code>x</code>	numeric matrix $A^{n \times f}$ with $n$ cells, and $f$ low-dimensional projections
<code>ks</code>	integer vector, $k$ values to consider
<code>B_gap</code>	integer, number of Monte Carlo ("bootstrap") samples taken when computing the Gap statistic (see documentation of function <code>clusGap</code> , R-package <code>cluster</code> )
<code>n_start</code> , <code>iter_max</code> , <code>kmeans_algorithm</code>	parameters for k-means clustering, see documentation of function <code>kmeans</code> , R-package <code>stats</code>
<code>cores</code>	integer, number of PC cores for parallel execution
<code>verbose</code>	logical, progress messages

**Details**

To compute the Gap statistic `get_k` adapts the algorithm of function `clustGap` from R-package `cluster` (version 2.1.3). For k-means clustering `get_k` uses the function `kmeans` implemented in R-package `stats` (version 4.2.0). For additional information see the respective documentations.

**Value**

`boot_obj`                    The results: k-means clustering solutions, the Gap statistic and WCSS

`gap_stats_summary, wcss_stats_summary`  
                               main results; Gap statistic and WCSS estimates. Means, standard errors and 95% confidence intervals are provided for each  $k$

`gap_stats, wcss_stats`  
                               intermediate results; Gap statistic and WCSS estimates for each  $k$  and bootstrap iteration  $b$

**Author(s)**

Simo Kitanovski <simo.kitanovski@uni-due.de>

**See Also**

`get_r`, `get_bubbletree_dummy`, `get_bubbletree_graph`, `get_bubbletree_kmeans`, `get_gini`, `get_gini_k`, `d_500`, `get_num_tiles`, `get_num_violins`, `get_cat_tiles`

**Examples**

```
# input data
data("d_500", package = "scBubbletree")
A <- d_500$A

b <- get_k(x = A,
          ks = 1:5,
          B_gap = 10,
          n_start = 100,
          iter_max = 200,
          kmeans_algorithm = "MacQueen",
          cores = 1,
          verbose = TRUE)

b$gap_stats_summary
```

---

<code>get_num_cell_tiles</code>	<i>Visualization of numeric features of individual cells using tile plots</i>
---------------------------------	---

---

**Description**

`get_num_cell_tiles` creates one heatmap from the cells in each bubble. The heatmap visualizes a gradient of the sorted (from high to low) values of a numeric feature (e.g. expression of a certain gene) among the cells of that bubble.

**Usage**

```
get_num_cell_tiles(btd,
                  f,
                  tile_bw = FALSE,
                  x_axis_name = "cells",
                  feature_name = "Feature",
                  rotate_x_axis_labels = TRUE)
```

**Arguments**

btd	bubbletree object
f	numeric vector, numeric cell feature
x_axis_name	character, x-axis title
feature_name	character, color legend title
rotate_x_axis_labels	logical, should the x-axis labels be shown horizontally (rotate_x_axis_labels = FALSE) or vertically (rotate_x_axis_labels = TRUE)
tile_bw	logical, tile grayscale (tile_bw = TRUE) vs. color (tile_bw = FALSE, default)

**Details**

`get_num_cell_tiles` uses two main inputs:

1. bubbletree object
2. numeric vector of a numeric cell feature.

The order of the cells used to generate the bubbletree (input 1.) should correspond to the order of cell features in input vector `f` (input 2.)

This function does the following procedure for each bubble: 1. sort and rank the cells in each bubble: rank = 1 for the cell with the highest `f` value, rank =  $n$  for the bubble with the lowest `f` value 2. draw a heatmap with `x=rank`, `y=bubble`, `tile-color=f`

**Value**

plot	ggplot2, tile plot
table	data.frame, raw data used to generate the plot

**Author(s)**

Simo Kitanovski <simo.kitanovski@uni-due.de>

**See Also**

`get_k`, `get_r`, `get_bubbletree_dummy`, `get_bubbletree_kmeans`, `get_bubbletree_graph`, `get_gini`, `get_gini_k`, `get_cat_tile`, `get_num_tiles`, `get_num_violins`, `d_500`, `d_ccl`

**Examples**

```
# input data
data("d_500", package = "scBubbletree")
A <- d_500$A
f <- as.vector(d_500$fs[,1])
```

```

b <- get_bubbletree_kmeans(x = A, k = 8)

g <- get_num_cell_tiles(btd = b, f = f)

b$tree|g$plot

```

get\_num\_tiles

*Visualization of numeric cell features using tile plots*

## Description

get\_num\_tiles creates tile plot to visualize a summary (e.g. mean, median or sum) of a numeric cell feature (e.g. gene expression of a specific gene) in each bubble of a bubbletree

## Usage

```

get_num_tiles(btd,
              fs,
              summary_function,
              round_digits = 2,
              tile_text_size = 3,
              tile_bw = FALSE,
              x_axis_name = "Feature",
              rotate_x_axis_labels = TRUE)

```

## Arguments

btd	bubbletree object
fs	numeric vector or matrix, numeric cell features
summary_function	character, "mean", "median" or "sum", "pct nonzero", "pct zero", summaries are allowed
round_digits	integer, number of decimal places to keep when showing the relative frequency of cells in each bubble
tile_text_size	integer, size of tile labels
x_axis_name	character, x-axis title
rotate_x_axis_labels	logical, should the x-axis labels be shown horizontally (rotate_x_axis_labels = FALSE) or vertically (rotate_x_axis_labels = TRUE)
tile_bw	logical, tile grayscale (tile_bw = TRUE) vs. color (tile_bw = FALSE, default)

## Details

get\_num\_tiles uses two main inputs:

1. bubbletree object
2. numeric vector or matrix of numeric cell features.

The order of the cells used to generat the bubbletree (input 1.) should correspond to the order of cells in the vector/matrix of numeric cell features (input 2.)

This function computes summaries of numeric cell feature in each bubble: 1. mean = mean of feature 2. median = median of feature 3. sum = sum of feature 4. pct nonzero = sum of cells with feature > 0 5. pct zero = sum of cells with feature = 0

Important note: NA and NULL values are omitted.

### Value

plot	ggplot2, tile plot
table	data.frame, raw data used to generate the plot

### Author(s)

Simo Kitanovski <simo.kitanovski@uni-due.de>

### See Also

get\_k, get\_r get\_bubbletree\_dummy, get\_bubbletree\_kmeans, get\_bubbletree\_graph, get\_gini, get\_gini\_k, get\_cat\_tile, get\_num\_violins, d\_500, d\_ccl

### Examples

```
# input data
data("d_500", package = "scBubbletree")
A <- d_500$A
fs <- d_500$fs

b <- get_bubbletree_kmeans(x = A, k = 8)

g <- get_num_tiles(btd = b,
                  fs = fs,
                  summary_function = "mean",
                  round_digits = 2,
                  tile_text_size = 3,
                  tile_bw = TRUE,
                  x_axis_name = "Gene expression",
                  rotate_x_axis_labels = TRUE)

b$tree|g$plot
```

---

get\_num\_violins

*Visualization of numeric cell features using violin plots*

---

### Description

get\_num\_violins creates violin plot to visualize the distribution of of numeric cell features (e.g. gene expressions) in each bubble of a bubbletree

### Usage

```
get_num_violins(btd,
                fs,
                x_axis_name = "Feature distribution",
                rotate_x_axis_labels = TRUE)
```

**Arguments**

btd	bubbletree object
fs	numeric vector or matrix, numeric cell features
x_axis_name	character, x-axis title
rotate_x_axis_labels	logical, should the x-axis labels be shown horizontally (rotate_x_axis_labels = FALSE) or vertically (rotate_x_axis_labels = TRUE)

**Details**

get\_num\_violins uses two main inputs:

1. bubbletree object
2. numeric vector or matrix of numeric cell features.

The order of the cells used to generat the bubbletree (input 1.) should correspond to the order of cells in the vector/matrix of numeric cell features (input 2.)

This function visualizes densities of numeric cell feature in the different bubble.

**Value**

plot	ggplot2, violin plot
table	data.frame, raw data used to generate the plot

**Author(s)**

Simo Kitanovski <simon.kitanovski@uni-due.de>

**See Also**

get\_k, get\_r get\_bubbletree\_dummy, get\_bubbletree\_kmeans, get\_bubbletree\_graph, get\_gini, get\_gini\_k, get\_cat\_tile, get\_num\_tiles, d\_500

**Examples**

```
# input data
data("d_500", package = "scBubbletree")
A <- d_500$A
fs <- d_500$fs

b <- get_bubbletree_graph(x = A, r = 0.8)

g <- get_num_violins(btd = b,
                    fs = fs,
                    x_axis_name = "Feature distribution",
                    rotate_x_axis_labels = TRUE)

b$tree|g$plot
```

---

get_r	<i>Finding optimal clustering resolution <math>r</math> and number of communities <math>k'</math></i>
-------	---

---

## Description

To perform Louvain clustering we must specify a clustering resolution  $r$ . Data-driven metrics, such as the Gap statistic or the within-cluster sum of squares (WCSS) can be used to infer appropriate  $r$  from the data. `get_r` computes the Gap statistic and WCSS for a vector of clustering resolutions  $rs$ .

## Usage

```
get_r(x,
      rs,
      B_gap = 20,
      n_start = 20,
      iter_max = 100,
      algorithm = "original",
      knn_k = 20,
      cores = 1,
      verbose = TRUE)
```

## Arguments

<code>x</code>	numeric matrix $A^{n \times f}$ with $n$ cells, and $f$ low-dimensional projections
<code>rs</code>	number vector, $r$ values to consider
<code>B_gap</code>	integer, number of Monte Carlo ("bootstrap") samples taken when computing the Gap statistic (see documentation of function <code>clusGap</code> , R-package <code>cluster</code> )
<code>n_start, iter_max</code>	parameters for Louvain clustering, see documentation of function <code>FindClusters</code> , R-package <code>Seurat</code>
<code>algorithm</code>	character, four clustering algorithms: 'original', 'LMR', 'SLM' and 'Leiden', see documentation of function <code>FindClusters</code> , R-package <code>Seurat</code>
<code>knn_k</code>	integer, defines $k$ for the $k$ -nearest neighbor algorithm, see documentation of function <code>FindClusters</code> , R-package <code>Seurat</code>
<code>cores</code>	integer, number of PC cores for parallel execution
<code>verbose</code>	logical, progress messages

## Details

To compute the Gap statistic `get_r` adapts the algorithm of function `clusGap` from R-package `cluster` (version 2.1.3). For Louvain clustering `get_r` uses the function `FindClusters` implemented in the R-package `Seurat`. For additional information see the respective documentations.

## Value

<code>boot_obj</code>	The results: k-means clustering solutions, the Gap statistic and WCSS
<code>gap_stats_summary, wcss_stats_summary</code>	main results; Gap statistic and WCSS estimates. Means, standard errors and 95% confidence intervals are provided for each $r$ and $k'$



gap\_stats, wcss\_stats

intermediate results; Gap statistic and WCSS estimates for each  $r$  and  $k'$  and bootstrap iteration  $b$

### Author(s)

Simo Kitanovski <simo.kitanovski@uni-due.de>

### See Also

get\_k, get\_bubbletree\_dummy, get\_bubbletree\_graph, get\_bubbletree\_kmeans, get\_gini, get\_gini\_k, d\_500, get\_num\_tiles, get\_num\_violins, get\_cat\_tiles, d\_ccl

### Examples

```
# input data
data("d_500", package = "scBubbletree")
A <- d_500$A
```

```
b <- get_r(x = A,
          rs = c(0.1, 0.5, 1),
          B_gap = 10,
          n_start = 20,
          iter_max = 100,
          algorithm = "original",
          cores = 1,
          verbose = TRUE)
```

```
b$gap_stats_summary
```

# Index

## \* datasets

d\_500, [4](#)

d\_ccl, [5](#)

compare\_bubbletrees, [3](#)

d\_500, [4](#)

d\_ccl, [5](#)

get\_bubbletree\_dummy, [6](#)

get\_bubbletree\_graph, [8](#)

get\_bubbletree\_kmeans, [11](#)

get\_cat\_tiles, [13](#)

get\_gini, [15](#)

get\_gini\_k, [16](#)

get\_k, [18](#)

get\_num\_cell\_tiles, [19](#)

get\_num\_tiles, [21](#)

get\_num\_violins, [22](#)

get\_r, [24](#)

scBubbletree (scBubbletree-package), [2](#)

scBubbletree-package, [2](#)