Package ‘scBubbletree’

May 13, 2024

Type Package

Title Quantitative visual exploration of scRNA-seq data

Version 1.6.0

Description scBubbletree is a quantitative method for visual exploration of scRNA-seq data. It preserves biologically meaningful properties of scRNA-seq data, such as local and global cell distances, as well as the density distribution of cells across the sample. scBubbletree is scalable and avoids the overplotting problem, and is able to visualize diverse cell attributes derived from multiomic single-cell experiments. Importantly, Importantly, scBubbletree is easy to use and to integrate with popular approaches for scRNA-seq data analysis.

License GPL-3 + file LICENSE

Depends R (>= 4.2.0)

Imports reshape2, future, future.apply, ape, scales, Seurat, ggplot2, ggtree, patchwork, proxy, methods, stats, base, utils

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 RELEASE_3_19
git_last_commit 1efa88b
git_last_commit_date 2024-04-30
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
Description

\texttt{d_500} is a list with 3 elements:
1. \texttt{A} = numeric matrix \(A^{500 \times 15}\) with \(n=500\) rows for PBMCs and \(f=15\) principal components.
2. \texttt{f} = character vector \(f\) of length 500. Each element in \(f\) represents the predicted cell type of a specific cell.
3. \texttt{fs} = numeric matrix containing normalized gene expressions of 12 marker genes in 500 cells.

Usage

\begin{verbatim}
data("d_500", package = "scBubbletree")
\end{verbatim}

Format

Format of \texttt{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

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

Examples

\begin{verbatim}
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)
\end{verbatim}
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

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

References


Examples

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

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

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

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 = 100, 
  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 bubble-tree. If B=0, 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. Maximum available number of cells are used for clusters that contain lower number of cells than N_eff
get_bubbletree_dummy

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
This function is similar to get_bubbletree_kmeans and get_bubbletree_graph. It skips the clustering step. See documentation of get_bubbletree_kmeans and get_bubbletree_graph.

Value
A
input x matrix

k
number of clusters

km
NULL

ph
boot_ph: bootstrap dendrograms $H_b$; main_ph: consensus dendrogram $\hat{H}$

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_meta
meta-data associated with the bubbletree

Author(s)
Simo Kitanovski <simo.kitanovski@uni-due.de>

See Also
get_k, get_r, get_bubbletree_kmeans, get_bubbletree_graph, 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
get_bubbletree_graph

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

db <- get_bubbletree_dummy(x = A,
    cs = cs,
    B = 100,
    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 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. 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

```r
get_bubbletree_graph(x,
    r,
    B = 100,
    N_eff = 200,
    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,
    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
get_bubbletree_graph

B
integer, number of bootstrap iterations to perform in order to generate bubble-tree. If B=0, 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. Maximum available number of cells are used for clusters that contain lower number of cells than N_eff

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 the algorithm performs the following steps:
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 Euclidean distances (in space \( A^{n \times f} \)) between all pairs of cells in cluster \( i \) and cluster \( j \)
4. compute mean Euclidean distance between cluster \( i \) and \( j \) and populate inter-cluster distance matrix \( D^{k \times k}_b \)
5. perform hierarchical clustering with average linkage based on \( D^{k \times k}_b \) to generate dendrogram \( H_b \)
6. compute average distance matrix \( \hat{D} \) and use is as input to build consensus hierarchical dendrogram (\( \hat{H} \); bubbletree) with average linkage
7. quantify branch robustness in \( \hat{H} \) count how many times each branch is found among bootstrap dendrograms (\( H_b \))
8. visualize the bubbletree (\( \hat{H} \)) with R-package ggtree
Special case: If B=0, then cluster centroids are used to compute inter-cluster distances and N_eff is ignored, i.e. all cells are used to compute centroids. This leads to computational efficiency, however, by doing so we lose information about the robustness of branches.

Value

<table>
<thead>
<tr>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
</tr>
<tr>
<td>k</td>
</tr>
<tr>
<td>r</td>
</tr>
<tr>
<td>ph</td>
</tr>
<tr>
<td>pair_dist</td>
</tr>
<tr>
<td>cluster</td>
</tr>
<tr>
<td>input_par</td>
</tr>
<tr>
<td>tree</td>
</tr>
<tr>
<td>tree_meta</td>
</tr>
</tbody>
</table>

Author(s)

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

See Also

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

Examples

```r
# 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 = 50,
                  hclust_method = "average",
                  hclust_distance = "euclidean",
                  cores = 1,
                  round_digits = 2,
                  show_simple_count = FALSE)

b$tree
```
get_bubbletree_kmeans  

*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*

```r
get_bubbletree_kmeans(x, 
  k, 
  B = 100, 
  N_eff = 200, 
  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. If B=0, 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. Maximum available number of cells are used for clusters that contain lower number of cells than N_eff
- `n_start, iter_max, kmeans_algorithm`: parameters for k-means clustering, see documentation of function k-means, R-package stats
get_bubbletree_kmeans

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 the algorithm performs the following steps:

1. In bootrap iteration \( b \) from 1 : \( B \)
2. draw up to \( N_{eff} \) number of cells at random from each cluster without replacement
3. compute Euclidean distances (in space \( A^{n\times f} \)) between pairs of cells in cluster \( i \) and cluster \( j \)
4. compute mean Euclidean distance between cluster \( i \) and \( j \) and populate inter-cluster distance matrix \( D_{b}^{k\times k} \)
5. perform hierarchical clustering with average linkage based on \( D_{b}^{k\times k} \) to generate dendrogram \( H_{b} \)
6. compute average distance matrix \( \hat{D} \) and use as input to build consensus hierarchical dendrogram (\( \hat{H} \); bubbletree) with average linkage
7. quantify branch robustness in \( \hat{H} \) count how many times each branch is found among bootstrap dendrograms (\( H_{b} \))
8. visualize the bubbletree (\( \hat{H} \)) with R-package ggtree

Special case: If \( B=0 \), then cluster centroids are used to compute inter-cluster distances and \( N_{eff} \) is ignored, i.e. all cells are used to compute centroids. This leads to computational efficiency, however, by doing so we lose information about the robustness of branches.

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: consensus dendrogram \( \hat{H} \)

\( pair_{dist} \) inter-cluster distances used to generate the dendrograms
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)
get_cat_tiles

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
the 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

```r
# 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
get_gini

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 bubletree, 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:

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

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

Value

<table>
<thead>
<tr>
<th>gi</th>
<th>Gini impurity of each bubble</th>
</tr>
</thead>
<tbody>
<tr>
<td>wgi</td>
<td>Weighted Gini impurity index of the bubletree</td>
</tr>
</tbody>
</table>

Author(s)

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

See Also

get_k, get_r, get_bubletree_kmeans, get_bubletree_dummy, get_bubletree_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

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.

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

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

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 bubble tree, 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:

\[
WGI = \frac{\sum_{i=1}^{k} GI_i n_i}{n},
\]

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

Value

- **gi_summary**: GI for each bubble of a clustering solution with clustering resolution \( k \) or \( r \)
- **wgi_summary**: WGI for each clustering solution with clustering resolution \( k \) or \( r \)

Author(s)

Simo Kitanovski &lt;simo.kitanovski@uni-due.de&gt;

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

```r
# 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 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

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

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 = "mean",
    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
**get_num_tiles**

- **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 generate 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

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

b <- get_bubbletree_kmeans(x = A,
                         k = 8,
                         N_eff = 100)

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",
```
get_num_violins

rotate_x_axis_labels = TRUE)

b$tree|g$plot

gtget_num_violins

Visualization of numeric cell features using violin plots

Description

gtget_num_violins creates violin plot to visualize the distribution 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

gtget_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 generate 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 <simo.kitanovski@uni-due.de>
get_r

Finding optimal clustering resolution $r$ and number of communities $k'$

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

```r
get_r(x, 
rs, 
B_gap = 20, 
n_start = 20, 
iter_max = 100, 
algorithm = "original", 
knn_k = 50, 
cores = 1, 
verbose = TRUE)
```

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

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

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

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

b$tree|g$plot
```
**Arguments**

- `x`: numeric matrix $A^{nxf}$ with $n$ cells, and $f$ low-dimensional projections
- `rs`: number vector, $r$ values to consider
- `B_gap`: integer, number of Monte Carlo ("bootstrap") samples taken when computing the Gap statistic (see documentation of function clusGap, R-package cluster)
- `n_start, iter_max`: parameters for Louvain clustering, 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
- `cores`: integer, number of PC cores for parallel execution
- `verbose`: logical, progress messages

**Details**

To compute the Gap statistic `get_r` adapts the algorithm of function clustGap from R-package cluster (version 2.1.3). For Louvain clustering `get_r` uses the function FindClusters implemented in R-package Seurat. 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 $r$ 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**

```r
# 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, 3
  d_ccl, 4

d_500, 3
d_ccl, 4

get_bubbletree_dummy, 5
get_bubbletree_graph, 7
get_bubbletree_kmeans, 10
get_cat_tiles, 12
get_gini, 14
get_gini_k, 16
get_k, 17
get_num_tiles, 19
get_num_violins, 21
get_r, 22

scBubbletree (scBubbletree-package), 2
scBubbletree-package, 2