

# Package: ClustAssess (via r-universe)

September 11, 2024

**Type** Package

**Title** Tools for Assessing Clustering

**Version** 0.3.1

**Maintainer** Arash Shahsavari <as3006@cam.ac.uk>

**Description** A set of tools for evaluating clustering robustness using proportion of ambiguously clustered pairs (Senbabaoglu et al. (2014) <[doi:10.1038/srep06207](https://doi.org/10.1038/srep06207)>), as well as similarity across methods and method stability using element-centric clustering comparison (Gates et al. (2019) <[doi:10.1038/s41598-019-44892-y](https://doi.org/10.1038/s41598-019-44892-y)>). Additionally, this package enables stability-based parameter assessment for graph-based clustering pipelines typical in single-cell data analysis.

**License** MIT + file LICENSE

**Encoding** UTF-8

**Imports** ggplot2, dplyr, fastcluster, rlang, Matrix, igraph, magrittr, Rcpp, methods, stats, foreach, doParallel, irlba, progress, reshape2, stringr, gtable, uwot

**RoxygenNote** 7.2.0

**LinkingTo** Rcpp

**Suggests** knitr, rmarkdown, e1071, dbscan, dendextend, Seurat, readr, patchwork

**URL** <https://github.com/Core-Bioinformatics/ClustAssess>

**BugReports** <https://github.com/Core-Bioinformatics/ClustAssess/issues>

**VignetteBuilder** knitr

**Repository** <https://core-bioinformatics.r-universe.dev>

**RemoteUrl** <https://github.com/core-bioinformatics/clustassess>

**RemoteRef** HEAD

**RemoteSha** 299c3b5ffd7a62832767cc8b743bb6e6e3f0ae1e

## Contents

consensus_cluster . . . . .	2
element_agreement . . . . .	4
element_consistency . . . . .	5
element_sim . . . . .	7
element_sim_elscore . . . . .	9
element_sim_matrix . . . . .	11
get_clustering_difference . . . . .	12
get_feature_stability . . . . .	14
get_nn_conn_comps . . . . .	15
get_nn_importance . . . . .	17
get_resolution_importance . . . . .	19
marker_overlap . . . . .	21
merge_partitions . . . . .	22
pac_convergence . . . . .	23
pac_landscape . . . . .	24
plot_clustering_difference_boxplot . . . . .	24
plot_clustering_difference_facet . . . . .	25
plot_connected_comps_evolution . . . . .	27
plot_feature_stability_boxplot . . . . .	28
plot_feature_stability_ecs_facet . . . . .	29
plot_feature_stability_ecs_incremental . . . . .	30
plot_feature_stability_mb_facet . . . . .	31
plot_k_n_partitions . . . . .	32
plot_k_resolution_corresp . . . . .	33
plot_n_neigh_ecs . . . . .	35
plot_n_neigh_k_correspondence . . . . .	36
<b>Index</b>	<b>37</b>

---

consensus_cluster	<i>Consensus Clustering and Proportion of Ambiguously Clustered Pairs</i>
-------------------	---

---

### Description

Calculate consensus clustering and proportion of ambiguously clustered pairs (PAC) with hierarchical clustering.

### Usage

```
consensus_cluster(
  x,
  k_min = 3,
  k_max = 100,
  n_reps = 100,
  p_sample = 0.8,
  p_feature = 1,
```

```

    p_minkowski = 2,
    dist_method = "euclidean",
    linkage = "complete",
    lower_lim = 0.1,
    upper_lim = 0.9,
    verbose = TRUE
)

```

### Arguments

x	A samples x features normalized data matrix.
k_min	The minimum number of clusters calculated.
k_max	The maximum number of clusters calculated.
n_reps	The total number of subsamplings and reclusterings of the data; this value needs to be high enough to ensure PAC converges; convergence can be assessed with <code>pac_convergence</code> .
p_sample	The proportion of samples included in each subsample.
p_feature	The proportion of features included in each subsample.
p_minkowski	The power of the Minkowski distance.
dist_method	The distance measure for the distance matrix used in <code>hclust</code> ; must be one of "euclidean", "maximum", "manhattan", "canberra", "binary" or "minkowski".
linkage	The linkage method used in <code>hclust</code> ; must be one of "ward.D", "ward.D2", "single", "complete", "average", "mcquitty", "median" or "centroid"
lower_lim	The lower limit for determining whether a pair is clustered ambiguously; the lower this value, the higher the PAC.
upper_lim	The upper limit for determining whether a pair is clustered ambiguously; the higher this value, the higher the PAC.
verbose	Logical value used for choosing to display a progress bar or not.

### Value

A data.frame with PAC values across iterations, as well as parameter values used when calling the method.

### References

- Monti, S., Tamayo, P., Mesirov, J., & Golub, T. (2003). Consensus clustering: a resampling-based method for class discovery and visualization of gene expression microarray data. *Machine learning*, 52(1), 91-118. <https://doi.org/10.1023/A:1023949509487>
- Senbabaoglu, Y., Michailidis, G., & Li, J. Z. (2014). Critical limitations of consensus clustering in class discovery. *Scientific reports*, 4(1), 1-13. <https://doi.org/10.1038/srep06207>

### Examples

```

pac.res = consensus_cluster(iris[,1:4], k_max=20)
pac_convergence(pac.res, k_plot=c(3,5,7,9))

```

---

element\_agreement      *Element-Wise Average Agreement Between a Set of Clusterings*

---

### Description

Inspect how consistently of a set of clusterings agree with a reference clustering by calculating their element-wise average agreement.

### Usage

```
element_agreement(
  reference_clustering,
  clustering_list,
  alpha = 0.9,
  r = 1,
  rescale_path_type = "max",
  ppr_implementation = "prpack",
  dist_rescaled = FALSE,
  row_normalize = TRUE,
  ncores = 1
)
```

### Arguments

**reference\_clustering**  
The reference clustering, that each clustering in `clustering_list` is compared to. It can be either:

- A numeric/character/factor vector of cluster labels for each element.
- A samples x clusters matrix/Matrix::Matrix of nonzero membership values.
- An hclust object.

**clustering\_list**  
The list of clustering results, each of which is either:

- A numeric/character/factor vector of cluster labels for each element.
- A samples x clusters matrix/Matrix::Matrix of nonzero membership values.
- An hclust object.

**alpha**  
A numeric giving the personalized PageRank damping factor; 1 - alpha is the restart probability for the PPR random walk.

**r**  
A numeric hierarchical scaling parameter.

**rescale\_path\_type**  
A string; rescale the hierarchical height by:

- "max" : the maximum path from the root.
- "min" : the minimum path form the root.
- "linkage" : use the linkage distances in the clustering.

ppr_implementation	Choose a implementation for personalized page-rank calculation: <ul style="list-style-type: none"> <li>• "prpack": use PPR algorithms in igraph.</li> <li>• "power_iteration": use power_iteration method.</li> </ul>
dist_rescaled	A logical: if TRUE, the linkage distances are linearly rescaled to be in-between 0 and 1.
row_normalize	Whether to normalize all rows in clustering_result so they sum to one before calculating ECS. It is recommended to set this to TRUE, which will lead to slightly different ECS values compared to clusim.
ncores	the number of parallel R instances that will run the code. If the value is set to 1, the code will be run sequentially.

**Value**

A vector containing the element-wise average agreement.

**References**

Gates, A. J., Wood, I. B., Hetrick, W. P., & Ahn, Y. Y. (2019). Element-centric clustering comparison unifies overlaps and hierarchy. *Scientific reports*, 9(1), 1-13. <https://doi.org/10.1038/s41598-019-44892-y>

**Examples**

```
# perform k-means clustering across 20 random seeds
reference.clustering = iris$Species
clustering.list = lapply(1:20, function(x) kmeans(iris[,1:4], centers=3)$cluster)
element_agreement(reference.clustering, clustering.list)
```

---

element\_consistency     *Element-Wise Consistency Between a Set of Clusterings*

---

**Description**

Inspect the consistency of a set of clusterings by calculating their element-wise clustering consistency (also known as element-wise frustration).

**Usage**

```
element_consistency(
  clustering_list,
  alpha = 0.9,
  r = 1,
  rescale_path_type = "max",
  ppr_implementation = "prpack",
  dist_rescaled = FALSE,
  row_normalize = TRUE,
  ncores = 1
)
```

**Arguments**

clustering_list	The list of clustering results, each of which is either: <ul style="list-style-type: none"> <li>• A numeric/character/factor vector of cluster labels for each element.</li> <li>• A samples x clusters matrix/Matrix::Matrix of nonzero membership values.</li> <li>• An hclust object.</li> </ul>
alpha	A numeric giving the personalized PageRank damping factor; 1 - alpha is the restart probability for the PPR random walk.
r	A numeric hierarchical scaling parameter.
rescale_path_type	A string; rescale the hierarchical height by: <ul style="list-style-type: none"> <li>• "max" : the maximum path from the root.</li> <li>• "min" : the minimum path form the root.</li> <li>• "linkage" : use the linkage distances in the clustering.</li> </ul>
ppr_implementation	Choose a implementation for personalized page-rank calculation: <ul style="list-style-type: none"> <li>• "prpack": use PPR algorithms in igraph.</li> <li>• "power_iteration": use power_iteration method.</li> </ul>
dist_rescaled	A logical: if TRUE, the linkage distances are linearly rescaled to be in-between 0 and 1.
row_normalize	Whether to normalize all rows in clustering_result so they sum to one before calculating ECS. It is recommended to set this to TRUE, which will lead to slightly different ECS values compared to clusim.
ncores	the number of parallel R instances that will run the code. If the value is set to 1, the code will be run sequentially.

**Value**

a vector containing the element-wise consistency.

**References**

Gates, A. J., Wood, I. B., Hetrick, W. P., & Ahn, Y. Y. (2019). Element-centric clustering comparison unifies overlaps and hierarchy. *Scientific reports*, 9(1), 1-13. <https://doi.org/10.1038/s41598-019-44892-y>

**Examples**

```
# cluster across 20 random seeds
clustering.list = lapply(1:20, function(x) kmeans(mtcars, centers=3)$cluster)
element_consistency(clustering.list)
```

**Description**

Calculates the average element-centric similarity between two clustering results

**Usage**

```
element_sim(
  clustering1,
  clustering2,
  alpha = 0.9,
  r_c11 = 1,
  rescale_path_type_c11 = "max",
  ppr_implementation_c11 = "prpack",
  dist_rescaled_c11 = FALSE,
  row_normalize_c11 = TRUE,
  r_c12 = 1,
  rescale_path_type_c12 = "max",
  ppr_implementation_c12 = "prpack",
  dist_rescaled_c12 = FALSE,
  row_normalize_c12 = TRUE
)
```

**Arguments**

- |                       |   |
|-----------------------|---|
| clustering1           | The first clustering result, which can be one of: <ul style="list-style-type: none"> <li>• A numeric/character/factor vector of cluster labels for each element.</li> <li>• A samples x clusters matrix/Matrix::Matrix of nonzero membership values.</li> <li>• An hclust object.</li> </ul>  |
| clustering2           | The second clustering result, which can be one of: <ul style="list-style-type: none"> <li>• A numeric/character/factor vector of cluster labels for each element.</li> <li>• A samples x clusters matrix/Matrix::Matrix of nonzero membership values.</li> <li>• An hclust object.</li> </ul> |
| alpha                 | A numeric giving the personalized PageRank damping factor; 1 - alpha is the restart probability for the PPR random walk.  |
| r_c11                 | A numeric hierarchical scaling parameter for the first clustering.  |
| rescale_path_type_c11 | A string; rescale the hierarchical height of the first clustering by: <ul style="list-style-type: none"> <li>• "max" : the maximum path from the root.</li> <li>• "min" : the minimum path form the root.</li> <li>• "linkage" : use the linkage distances in the clustering.</li> </ul>      |

ppr\_implementation\_cl1

Choose a implementation for personalized page-rank calculation for the first clustering:

- "prpack": use PPR algorithms in igraph.
- "power\_iteration": use power\_iteration method.

dist\_rescaled\_cl1

A logical: if TRUE, the linkage distances of the first clustering are linearly rescaled to be in-between 0 and 1.

row\_normalize\_cl1

Whether to normalize all rows in the first clustering so they sum to one before calculating ECS. It is recommended to set this to TRUE, which will lead to slightly different ECS values compared to clusim.

r\_cl2

A numeric hierarchical scaling parameter for the second clustering.

rescale\_path\_type\_cl2

A string; rescale the hierarchical height of the second clustering by:

- "max" : the maximum path from the root.
- "min" : the minimum path form the root.
- "linkage" : use the linkage distances in the clustering.

ppr\_implementation\_cl2

Choose a implementation for personalized page-rank calculation for the second clustering:

- "prpack": use PPR algorithms in igraph.
- "power\_iteration": use power\_iteration method.

dist\_rescaled\_cl2

A logical: if TRUE, the linkage distances of the second clustering are linearly rescaled to be in-between 0 and 1.

row\_normalize\_cl2

Whether to normalize all rows in the second clustering so they sum to one before calculating ECS. It is recommended to set this to TRUE, which will lead to slightly different ECS values compared to clusim.

## Value

The average element-wise similarity between the two Clusterings.

## Examples

```
km.res = kmeans(mtcars, centers=3)$cluster
hc.res = hclust(dist(mtcars))
element_sim(km.res, hc.res)
```



---

element\_sim\_elscore     *The Element-Centric Clustering Similarity for each Element*

---

## Description

Calculates the element-wise element-centric similarity between two clustering results.

## Usage

```
element_sim_elscore(
  clustering1,
  clustering2,
  alpha = 0.9,
  r_c11 = 1,
  rescale_path_type_c11 = "max",
  ppr_implementation_c11 = "prpack",
  dist_rescaled_c11 = FALSE,
  row_normalize_c11 = TRUE,
  r_c12 = 1,
  rescale_path_type_c12 = "max",
  ppr_implementation_c12 = "prpack",
  dist_rescaled_c12 = FALSE,
  row_normalize_c12 = TRUE
)
```

## Arguments

- |                       |   |
|-----------------------|---|
| clustering1           | The first clustering result, which can be one of: <ul style="list-style-type: none"> <li>• A numeric/character/factor vector of cluster labels for each element.</li> <li>• A samples x clusters matrix/Matrix::Matrix of nonzero membership values.</li> <li>• An hclust object.</li> </ul>  |
| clustering2           | The second clustering result, which can be one of: <ul style="list-style-type: none"> <li>• A numeric/character/factor vector of cluster labels for each element.</li> <li>• A samples x clusters matrix/Matrix::Matrix of nonzero membership values.</li> <li>• An hclust object.</li> </ul> |
| alpha                 | A numeric giving the personalized PageRank damping factor; 1 - alpha is the restart probability for the PPR random walk.  |
| r_c11                 | A numeric hierarchical scaling parameter for the first clustering.  |
| rescale_path_type_c11 | A string; rescale the hierarchical height of the first clustering by: <ul style="list-style-type: none"> <li>• "max" : the maximum path from the root.</li> <li>• "min" : the minimum path form the root.</li> <li>• "linkage" : use the linkage distances in the clustering.</li> </ul>      |

`ppr_implementation_c11`  
 Choose a implementation for personalized page-rank calculation for the first clustering:

- "prpack": use PPR algorithms in igraph.
- "power\_iteration": use power\_iteration method.

`dist_rescaled_c11`  
 A logical: if TRUE, the linkage distances of the first clustering are linearly rescaled to be in-between 0 and 1.

`row_normalize_c11`  
 Whether to normalize all rows in the first clustering so they sum to one before calculating ECS. It is recommended to set this to TRUE, which will lead to slightly different ECS values compared to clusim.

`r_c12`  
 A numeric hierarchical scaling parameter for the second clustering.

`rescale_path_type_c12`  
 A string; rescale the hierarchical height of the second clustering by:

- "max" : the maximum path from the root.
- "min" : the minimum path form the root.
- "linkage" : use the linkage distances in the clustering.

`ppr_implementation_c12`  
 Choose a implementation for personalized page-rank calculation for the second clustering:

- "prpack": use PPR algorithms in igraph.
- "power\_iteration": use power\_iteration method.

`dist_rescaled_c12`  
 A logical: if TRUE, the linkage distances of the second clustering are linearly rescaled to be in-between 0 and 1.

`row_normalize_c12`  
 Whether to normalize all rows in the second clustering so they sum to one before calculating ECS. It is recommended to set this to TRUE, which will lead to slightly different ECS values compared to clusim.

**Value**

Vector of element-centric similarity between the two clusterings for each element.

**References**

Gates, A. J., Wood, I. B., Hetrick, W. P., & Ahn, Y. Y. (2019). Element-centric clustering comparison unifies overlaps and hierarchy. *Scientific reports*, 9(1), 1-13. <https://doi.org/10.1038/s41598-019-44892-y>

**Examples**

```
km.res = kmeans(iris[,1:4], centers=8)$cluster
hc.res = hclust(dist(iris[,1:4]))
element_sim_elscore(km.res, hc.res)
```

---

element\_sim\_matrix      *Pairwise Comparison of Clusterings*

---

### Description

Compare a set of clusterings by calculating their pairwise average element-centric clustering similarities.

### Usage

```
element_sim_matrix(
  clustering_list,
  output_type = "matrix",
  alpha = 0.9,
  r = 1,
  rescale_path_type = "max",
  ppr_implementation = "prpack",
  dist_rescaled = FALSE,
  row_normalize = TRUE,
  ncores = 1
)
```

### Arguments

**clustering\_list**      The list of clustering results, each of which is either:

- A numeric/character/factor vector of cluster labels for each element.
- A samples x clusters matrix/Matrix::Matrix of nonzero membership values.
- An hclust object.

**output\_type**      A string specifying whether the output should be a matrix or a data.frame.

**alpha**      A numeric giving the personalized PageRank damping factor; 1 - alpha is the restart probability for the PPR random walk.

**r**      A numeric hierarchical scaling parameter.

**rescale\_path\_type**      A string; rescale the hierarchical height by:

- "max" : the maximum path from the root.
- "min" : the minimum path form the root.
- "linkage" : use the linkage distances in the clustering.

**ppr\_implementation**      Choose a implementation for personalized page-rank calculation:

- "prpack": use PPR algorithms in igraph.
- "power\_iteration": use power\_iteration method.

**dist\_rescaled**      A logical: if TRUE, the linkage distances are linearly rescaled to be in-between 0 and 1.

row_normalize	Whether to normalize all rows in clustering_result so they sum to one before calculating ECS. It is recommended to set this to TRUE, which will lead to slightly different ECS values compared to clusim.
ncores	the number of parallel R instances that will run the code. If the value is set to 1, the code will be run sequentially.

**Value**

A matrix or data.frame containing the pairwise ECS values.

**References**

Gates, A. J., Wood, I. B., Hetrick, W. P., & Ahn, Y. Y. (2019). Element-centric clustering comparison unifies overlaps and hierarchy. *Scientific reports*, 9(1), 1-13. <https://doi.org/10.1038/s41598-019-44892-y>

**Examples**

```
# cluster across 20 random seeds
clustering.list = lapply(1:20, function(x) kmeans(mtcars, centers=3)$cluster)
element_sim_matrix(clustering.list, output_type="matrix")
```

---

get\_clustering\_difference

*Graph Clustering Method Stability*

---

**Description**

Evaluates the stability of different graph clustering methods in the clustering pipeline. The method will iterate through different values of the resolution parameter and compare, using the EC Consistency score, the partitions obtained at different seeds.

**Usage**

```
get_clustering_difference(
  graph_adjacency_matrix,
  resolution,
  n_repetitions = 100,
  seed_sequence = NULL,
  ecs_thresh = 1,
  ncores = 1,
  algorithm = 1:4,
  verbose = TRUE
)
```

**Arguments**

graph_adjacency_matrix	A square adjacency matrix based on which an igraph object will be built.
resolution	A sequence of resolution values.
n_repetitions	The number of repetitions of applying the pipeline with different seeds; ignored if seed_sequence is provided by the user.
seed_sequence	A custom seed sequence; if the value is NULL, the sequence will be built starting from 1 with a step of 100.
ecs_thresh	The ECS threshold used for merging similar clusterings.
ncores	The number of parallel R instances that will run the code. If the value is set to 1, the code will be run sequentially.
algorithm	An index or a list of indexes indicating which community detection algorithm will be used: Louvain (1), Louvain refined (2), SLM (3) or Leiden (4). More details can be found in the Seurat's FindClusters function.
verbose	Boolean value used for displaying the progress bar.

**Value**

A list having two fields:

- all - a list that contains, for each clustering method and each resolution value, the EC consistency between the partitions obtained by changing the seed
- filtered - similar to all, but for each configuration, we determine the number of clusters that appears the most and use only the partitions with this size

**Examples**

```
set.seed(2021)
# create an artificial expression matrix
expr_matrix = matrix(runif(100*10), nrow = 100)
rownames(expr_matrix) = as.character(1:100)

adj_matrix = Seurat::FindNeighbors(expr_matrix,
  k.param = 10,
  nn.method = "rann",
  verbose = FALSE,
  compute.SNN = FALSE)$nn
clust_diff_obj = get_clustering_difference(graph_adjacency_matrix = adj_matrix,
  resolution = c(0.5, 1),
  n_repetitions = 10,
  algorithm = 1:2,
  verbose = FALSE)
plot_clustering_difference_boxplot(clust_diff_obj)
```

---

get\_feature\_stability *Evaluate Feature Set Stability*

---

### Description

Evaluate the stability of clusterings obtained based on incremental subsets of a given feature set.

### Usage

```
get_feature_stability(
  data_matrix,
  feature_set,
  steps,
  feature_type,
  n_repetitions = 100,
  seed_sequence = NULL,
  graph_reduction_type = "PCA",
  npcs = 30,
  ecs_thresh = 1,
  ncores = 1,
  algorithm = 4,
  ...
)
```

### Arguments

data_matrix	A data matrix having the features on the rows and the observations on the columns.
feature_set	A set of feature names that can be found on the rownames of the data matrix.
steps	Vector containing the sizes of the subsets; negative values will be interpreted as using all features.
feature_type	A name associated to the feature_set.
n_repetitions	The number of repetitions of applying the pipeline with different seeds; ignored if seed_sequence is provided by the user.
seed_sequence	A custom seed sequence; if the value is NULL, the sequence will be built starting from 1 with a step of 100.
graph_reduction_type	The graph reduction type, denoting if the graph should be built on either the PCA or the UMAP embedding.
npcs	The number of principal components.
ecs_thresh	The ECS threshold used for merging similar clusterings.
ncores	The number of parallel R instances that will run the code. If the value is set to 1, the code will be run sequentially.

algorithm      An index indicating which community detection algorithm will be used: Louvain (1), Louvain refined (2), SLM (3) or Leiden (4). More details can be found in the Seurat's FindClusters function.

...              additional arguments passed to the umap method.

### Value

A list having one field associated with a step value. Each step contains a list with three fields:

- ecc - the EC-Consistency of the partitions obtained on all repetitions
- embedding - one UMAP embedding generated on the feature subset
- most\_frequent\_partition - the most common partition obtained across repetitions

### Note

The algorithm assumes that the feature\_set is already sorted when performing the subsetting. For example, if the user wants to analyze highly variable feature set, they should provide them sorted by their variability.

### Examples

```
set.seed(2021)
# create an artificial expression matrix
expr_matrix = matrix(c(runif(100*10), runif(100*10, min = 3, max = 4)), nrow = 200, byrow = TRUE)
rownames(expr_matrix) = as.character(1:200)
colnames(expr_matrix) = paste("feature", 1:10)

feature_stability_result = get_feature_stability(data_matrix = t(expr_matrix),
  feature_set = colnames(expr_matrix),
  feature_type = "feature_name",
  steps = 5,
  npcs = 2,
  n_repetitions = 10,
  algorithm = 1,
  # the following parameters are used by the umap function and are not mandatory
  n_neighbors = 3,
  approx_pow = TRUE,
  n_epochs = 0,
  init = "random",
  min_dist = 0.3)
plot_feature_stability_boxplot(feature_stability_result)
```

**Description**

One of the steps in the clustering pipeline is building a k-nearest neighbor graph on a reduced-space embedding. This method assesses the relationship between different number of nearest neighbors and the connectivity of the graph. In the context of graph clustering, the number of connected components can be used as a lower bound for the number of clusters. The calculations are performed multiple times by changing the seed at each repetition.

**Usage**

```
get_nn_conn_comps(
  object,
  n_neigh_sequence,
  config_name = "",
  n_repetitions = 100,
  seed_sequence = NULL,
  graph_reduction_type = "UMAP",
  transpose = (graph_reduction_type == "PCA"),
  ncores = 1,
  ...
)
```

**Arguments**

<code>object</code>	A data matrix. If the graph reduction type is PCA, the object should be an expression matrix, with features on rows and observations on columns; in the case of UMAP, the user could also provide a matrix associated to a PCA embedding. See also the transpose argument.
<code>n_neigh_sequence</code>	A sequence of the number of nearest neighbors.
<code>config_name</code>	User specified string that uniquely describes the embedding characteristics.
<code>n_repetitions</code>	The number of repetitions of applying the pipeline with different seeds; ignored if <code>seed_sequence</code> is provided by the user.
<code>seed_sequence</code>	A custom seed sequence; if the value is <code>NULL</code> , the sequence will be built starting from 1 with a step of 100.
<code>graph_reduction_type</code>	The graph reduction type, denoting if the graph should be built on either the PCA or the UMAP embedding.
<code>transpose</code>	Logical: whether the input object will be transposed or not. Set to <code>FALSE</code> if the input is an observations X features matrix, and set to <code>TRUE</code> if the input is a features X observations matrix.
<code>ncores</code>	The number of parallel R instances that will run the code. If the value is set to 1, the code will be run sequentially.
<code>...</code>	Additional arguments passed to the <code>'irlba::irlba'</code> or the <code>'uwot::umap'</code> method, depending on the value of <code>graph_reduction_type</code> .



**Value**

A list having one field associated with a number of nearest neighbors. Each value contains an array of the number of connected components obtained on the specified number of repetitions.

**Examples**

```
set.seed(2021)
# create an artificial expression matrix
expr_matrix = matrix(c(runif(50*10), runif(50*10, min = 1, max = 2)), nrow = 100, byrow = TRUE)
rownames(expr_matrix) = as.character(1:100)

# the graph reduction type is PCA, so we can provide the expression matrix as argument
nn_conn_comps_obj = get_nn_conn_comps(object = expr_matrix,
  n_neigh_sequence = c(2,3,5),
  config_name = "example_config",
  n_repetitions = 10,
  graph_reduction_type = "PCA",
  transpose = FALSE,
  # the following parameter is used by the irlba function and is not mandatory
  nv = 3)
plot_connected_comps_evolution(nn_conn_comps_obj)
```

---

get_nn_importance	<i>Assess Graph Building Parameters</i>
-------------------	---

---

**Description**

Evaluates clustering stability when changing the values of different parameters involved in the graph building step, namely the base embedding, the graph type and the number of neighbours.

**Usage**

```
get_nn_importance(
  object,
  n_neigh_sequence,
  n_repetitions = 100,
  seed_sequence = NULL,
  graph_reduction_type = "PCA",
  ecs_thresh = 1,
  ncores = 1,
  transpose = (graph_reduction_type == "PCA"),
  graph_type = 2,
  algorithm = 4,
  ...
)
```

**Arguments**

<code>object</code>	The data matrix. If the graph reduction type is PCA, the object should be an expression matrix, with features on rows and observations on columns; in the case of UMAP, the user could also provide a matrix associated to a PCA embedding. See also the <code>transpose</code> argument.
<code>n_neigh_sequence</code>	A sequence of the number of nearest neighbours.
<code>n_repetitions</code>	The number of repetitions of applying the pipeline with different seeds; ignored if <code>seed_sequence</code> is provided by the user.
<code>seed_sequence</code>	A custom seed sequence; if the value is NULL, the sequence will be built starting from 1 with a step of 100.
<code>graph_reduction_type</code>	The graph reduction type, denoting if the graph should be built on either the PCA or the UMAP embedding.
<code>ecs_thresh</code>	The ECS threshold used for merging similar clusterings.
<code>ncores</code>	The number of parallel R instances that will run the code. If the value is set to 1, the code will be run sequentially.
<code>transpose</code>	Logical: whether the input object will be transposed or not. Set to FALSE if the input is an observations X features matrix, and set to TRUE if the input is a features X observations matrix.
<code>graph_type</code>	Argument indicating whether the graph should be unweighted (0), weighted (1) or both (2).
<code>algorithm</code>	An index indicating which community detection algorithm will be used: Louvain (1), Louvain refined (2), SLM (3) or Leiden (4). More details can be found in the Seurat's <code>FindClusters</code> function.
<code>...</code>	Additional arguments passed to the <code>irlba::irlba</code> or the <code>uwot::umap</code> method, depending on the value of <code>graph_reduction_type</code> .

**Value**

A list having three fields:

- `n_neigh_k_corresp` - list containing the number of the clusters obtained by running the pipeline multiple times with different seed, number of neighbors and graph type (weighted vs unweighted)
- `n_neigh_ec_consistency` - list containing the EC consistency of the partitions obtained at multiple runs when changing the number of neighbors or the graph type
- `n_different_partitions` - the number of different partitions obtained by each number of neighbors

**Examples**

```
set.seed(2021)
# create an artificial expression matrix
expr_matrix = matrix(c(runif(100*10), runif(100*10, min=5, max=6)), nrow = 200)
```

```

rownames(expr_matrix) = as.character(1:200)

nn_importance_obj = get_nn_importance(object = expr_matrix,
  n_neigh_sequence = c(10,15,20),
  n_repetitions = 10,
  graph_reduction_type = "PCA",
  algorithm = 1,
  transpose = FALSE, # the matrix is already observations x features, so we won't transpose it
  # the following parameter is used by the irlba function and is not mandatory
  nv = 2)
plot_n_neigh_ecs(nn_importance_obj)

```

---

get\_resolution\_importance

*Evaluate Stability Across Resolution, Number of Neighbors, and Graph Type*

---

### Description

Perform a grid search over the resolution, number of neighbors and graph type.

### Usage

```

get_resolution_importance(
  embedding,
  resolution,
  n_neigh,
  n_repetitions = 100,
  seed_sequence = NULL,
  clustering_method = 4,
  graph_type = 0,
  object_name = NULL,
  ecs_thresh = 1,
  ncores = 1
)

```

### Arguments

embedding	The base embedding for the graph construction.
resolution	A sequence of resolution values.
n_neigh	A value or a sequence of number of neighbors used for graph construction.
n_repetitions	The number of repetitions of applying the pipeline with different seeds; ignored if seed_sequence is provided by the user.
seed_sequence	A custom seed sequence; if the value is NULL, the sequence will be built starting from 1 with a step of 100.

clustering_method	An index or a list of indexes indicating which community detection algorithm will be used: Louvain (1), Louvain refined (2), SLM (3) or Leiden (4). More details can be found in the Seurat's FindClusters function.
graph_type	Argument indicating whether the graph should be unweighted (0), weighted (1) or both (2).
object_name	User specified string that uniquely describes the embedding characteristics.
ecs_thresh	The ECS threshold used for merging similar clusterings.
ncores	The number of parallel R instances that will run the code. If the value is set to 1, the code will be run sequentially.

### Value

A list having two fields:

- `split_by_resolution`: A five-level list. The hierarchy is as follows:
  - the configuration name: concatenation between the object name provided by the user, the number of neighbors, the graph type and the clustering method
  - the resolution value  $\gamma$
  - the number of clusters  $k$  that can be obtained using the specified resolution
  - a `partitions` field containing the partitions obtained with resolution  $\gamma$  and have  $k$  clusters along with a `ecc` field that contains the Element-Centric Consistency of the partition list
  - the structure of a partition, which consists in having a `mb` field with the flat membership vector, `freq` denoting its frequency and `seed`, that is the seed used to obtain this partition in this configuration.
- `split_by_k`: has a similar structure, but the resolution level is removed. The partitions obtained in a configuration with the same number of clusters will be merged into the same list.

### Examples

```
set.seed(2021)
# create an artificial expression matrix
expr_matrix = matrix(runif(500*10), nrow = 500)

# get the PCA embedding of the data
pca_embedding = irlba::irlba(expr_matrix, nv = 2)
pca_embedding = pca_embedding$u %*% diag(pca_embedding$d)
rownames(pca_embedding) = as.character(1:500)

# run the function on the pca embedding
resolution_result = get_resolution_importance(embedding = pca_embedding,
  resolution = c(0.8, 1),
  n_neigh = c(5, 7),
  n_repetitions = 5,
  clustering_method = 1,
  graph_type = 2,
  object_name = "name_example")

plot_k_resolution_corresp(resolution_result)
```

---

marker_overlap	<i>Cell-Wise Marker Gene Overlap</i>
----------------	--------------------------------------

---

**Description**

Calculates the per-cell overlap of previously calculated marker genes.

**Usage**

```
marker_overlap(
  markers1,
  markers2,
  clustering1,
  clustering2,
  n = 25,
  overlap_type = "jsi",
  rank_by = "-p_val",
  use_sign = TRUE
)
```

**Arguments**

markers1	The first data frame of marker genes, must contain columns called 'gene' and 'cluster'.
markers2	The second data frame of marker genes, must contain columns called 'gene' and 'cluster'.
clustering1	The first vector of cluster assignments.
clustering2	The second vector of cluster assignments.
n	The number of top n markers (ranked by rank_by) to use when calculating the overlap.
overlap_type	The type of overlap to calculated: must be one of 'jsi' for Jaccard similarity index and 'intersect' for intersect size.
rank_by	A character string giving the name of the column to rank marker genes by. Note the sign here: to rank by lowest p-value, preface the column name with a minus sign; to rank by highest value, where higher value indicates more discriminative genes (for example power in the ROC test), no sign is needed.
use_sign	A logical: should the sign of markers match for overlap calculations? So a gene must be a positive or a negative marker in both clusters being compared. If TRUE, markers1 and markers2 must have a 'avg_logFC' or 'avg_log2FC' column, from which the sign of the DE will be extracted.

**Value**

A vector of the marker gene overlap per cell.

**Examples**

```

suppressWarnings({
  set.seed(1234)
  library(Seurat)
  data('pbmc_small')

  # cluster with Louvain algorithm
  pbmc_small = FindClusters(pbmc_small, resolution=0.8, verbose=FALSE)

  # cluster with k-means
  pbmc.pca = Embeddings(pbmc_small, 'pca')
  pbmc_small@meta.data$kmeans_clusters = kmeans(pbmc.pca, centers=3)$cluster

  # compare the markers
  Idents(pbmc_small) = pbmc_small@meta.data$seurat_clusters
  louvain.markers = FindAllMarkers(pbmc_small,
                                   logfc.threshold=1,
                                   test.use='t',
                                   verbose=FALSE)

  Idents(pbmc_small) = pbmc_small@meta.data$kmeans_clusters
  kmeans.markers = FindAllMarkers(pbmc_small,
                                   logfc.threshold=1,
                                   test.use='t',
                                   verbose=FALSE)

  pbmc_small@meta.data$jsi = marker_overlap(louvain.markers, kmeans.markers,
                                             pbmc_small@meta.data$seurat_clusters, pbmc_small@meta.data$kmeans_clusters)

  # which cells have the same markers, regardless of clustering?
  FeaturePlot(pbmc_small, 'jsi')
})

```

---

merge\_partitions

*Merge Partitions*


---

**Description**

Merge flat disjoint clusterings whose pairwise ECS score is above a given threshold. The merging is done using a complete linkage approach.

**Usage**

```
merge_partitions(partition_list, ecs_thresh = 1, ncores = 1, order = TRUE)
```

**Arguments**

`partition_list` A list of flat disjoint membership vectors.  
`ecs_thresh` A numeric: the ecs threshold.

ncores	The number of parallel R instances that will run the code. If the value is set to 1, the code will be run sequentially.
order	A logical: if TRUE, order the partitions based on their frequencies.

**Value**

a list of the merged partitions

**Examples**

```
initial_list = list(c(1,1,2), c(2,2,2), c('B','B','A'))
merge_partitions(initial_list, 0.99)
```

---

pac_convergence	<i>PAC Convergence Plot</i>
-----------------	-----------------------------

---

**Description**

Plot PAC across iterations for a set of k to assess convergence.

**Usage**

```
pac_convergence(pac_res, k_plot)
```

**Arguments**

pac_res	The data.frame output by consensus_cluster.
k_plot	A vector with values of k to plot.

**Value**

A ggplot2 object with the convergence plot. Convergence has been reached when the lines flatten out across k\_plot values.

**Examples**

```
pac.res = consensus_cluster(iris[,1:4], k_max=20)
pac_convergence(pac.res, k_plot=c(3,5,7,9))
```

---

pac\_landscape                    *PAC Landscape Plot*

---

### Description

Plot final PAC values across range of k to find optimal number of clusters.

### Usage

```
pac_landscape(pac_res, n_shade = max(pac_res$iteration)/5)
```

### Arguments

pac_res	The data.frame output by consensus_cluster.
n_shade	The PAC values across the last n_shade iterations will be shaded to illustrate the how stable the PAC score is.

### Value

A ggplot2 object with the final PAC vs k plot. A local minimum in the landscape indicates an especially stable value of k.

### Examples

```
pac.res = consensus_cluster(iris[,1:4], k_max=20)
pac_landscape(pac.res)
```

---

plot\_clustering\_difference\_boxplot  
*Clustering Method Stability Boxplot*

---

### Description

Display EC consistency across clustering method and resolution values. The 'filtered' field of the object returned by the 'get\_clustering\_difference\_object' method is used. Above each boxplot, the number of clusters is displayed.

### Usage

```
plot_clustering_difference_boxplot(
  clustering_difference_object,
  text_size = 3,
  dodge_width = 0.9,
  boxplot_width = 0.4
)
```



**Arguments**

clustering_difference_object	An object returned by the 'get_clustering_difference_object' method.
text_size	The size of the labels above boxplots.
dodge_width	Used for adjusting the horizontal position of the boxplot; the value will be passed to the 'width' argument of the 'ggplot2::position_dodge' method.
boxplot_width	Used for adjusting the width of the boxplots; the value will be passed to the 'width' argument of the 'ggplot2::geom_boxplot' method.

**Value**

A ggplot2 object with the EC consistency distributions. Higher consistency indicates a more stable clustering.

**Examples**

```
set.seed(2021)
# create an artificial expression matrix
expr_matrix = matrix(runif(500*10), nrow = 500)
rownames(expr_matrix) = as.character(1:500)

adj_matrix = Seurat::FindNeighbors(expr_matrix,
  k.param = 10,
  nn.method = "rann",
  verbose = FALSE,
  compute.SNN = FALSE)$nn
clust_diff_obj = get_clustering_difference(graph_adjacency_matrix = adj_matrix,
  resolution = c(0.5, 1),
  n_repetitions = 10,
  algorithm = 1:2,
  verbose = FALSE)
plot_clustering_difference_boxplot(clust_diff_obj)
```

---

```
plot_clustering_difference_facet
```

*Clustering Method Stability Facet Plot*

---

**Description**

Display the distribution of the EC consistency for each clustering method and each resolution value on a given embedding. The 'all' field of the object returned by the 'get\_clustering\_difference\_object' method is used.

**Usage**

```
plot_clustering_difference_facet(
  clustering_difference_object,
  embedding,
  low_limit = 0,
  high_limit = 1,
  grid = TRUE
)
```

**Arguments**

clustering_difference_object	An object returned by the 'get_clustering_difference_object' method.
embedding	An embedding (only the first two dimensions will be used for visualization).
low_limit	The lowest value of ECC that will be displayed on the embedding.
high_limit	The highest value of ECC that will be displayed on the embedding.
grid	Boolean value indicating whether the facet should be a grid (where each row is associated with a resolution value and each column with a clustering method) or a wrap.

**Value**

A ggplot2 object.

**Examples**

```
set.seed(2021)
# create an artificial expression matrix
expr_matrix = matrix(c(runif(250*10), runif(250*10, min = 5, max = 7)), nrow = 500)
rownames(expr_matrix) = as.character(1:500)

pca_embedding = irlba::irlba(expr_matrix, nv = 2)
pca_embedding = pca_embedding$u %*% diag(pca_embedding$d)
rownames(pca_embedding) = as.character(1:500)

adj_matrix = Seurat::FindNeighbors(pca_embedding,
  k.param = 10,
  nn.method = "rann",
  verbose = FALSE,
  compute.SNN = FALSE)$nn
clust_diff_obj = get_clustering_difference(graph_adjacency_matrix = adj_matrix,
  resolution = c(0.5, 1),
  n_repetitions = 10,
  algorithm = 1:2,
  verbose = FALSE)
plot_clustering_difference_facet(clust_diff_obj, pca_embedding)
```

---

`plot_connected_comps_evolution`*Relationship Between Number of Nearest Neighbors and Graph Connectivity*

---

**Description**

Display the distribution of the number connected components obtained for each number of neighbors across random seeds.

**Usage**

```
plot_connected_comps_evolution(nn_conn_comps_object)
```

**Arguments**

`nn_conn_comps_object`

An object or a concatenation of objects returned by the ‘get\_nn\_conn\_comps’ method.

**Value**

A ggplot2 object with boxplots for the connected component distributions.

**Note**

The number of connected components is displayed on a logarithmic scale.

**Examples**

```
set.seed(2021)
# create an artificial expression matrix
expr_matrix = matrix(c(runif(50*10), runif(50*10, min = 1, max = 2)), nrow = 100, byrow = TRUE)
rownames(expr_matrix) = as.character(1:100)

# the graph reduction type is PCA, so we can provide the expression matrix as argument
nn_conn_comps_obj = get_nn_conn_comps(object = expr_matrix,
  n_neigh_sequence = c(2,3,5),
  config_name = "example_config",
  n_repetitions = 10,
  graph_reduction_type = "PCA",
  transpose = FALSE,
  # the following parameter is used by the irlba function and is not mandatory
  nv = 3)
plot_connected_comps_evolution(nn_conn_comps_obj)
```

---

```
plot_feature_stability_boxplot
```

*Feature Stability Boxplot*

---

### Description

Display EC consistency for each feature set and for each step. Above each boxplot there is a number representing the step (or the size of the subset)

### Usage

```
plot_feature_stability_boxplot(
  feature_object_list,
  text_size = 4,
  boxplot_width = 0.4,
  dodge_width = 0.7
)
```

### Arguments

feature_object_list	An object or a concatenation of objects returned by the ‘get_feature_stability’ method
text_size	The size of the labels above boxplots
boxplot_width	Used for adjusting the width of the boxplots; the value will be passed to the ‘width’ argument of the ‘ggplot2::geom_boxplot’ method.
dodge_width	Used for adjusting the horizontal position of the boxplot; the value will be passed to the ‘width’ argument of the ‘ggplot2::position_dodge’ method.

### Value

A ggplot2 object.

### Examples

```
set.seed(2021)
# create an artificial expression matrix
expr_matrix = matrix(c(runif(100*10), runif(100*10, min = 3, max = 4)), nrow = 200, byrow = TRUE)
rownames(expr_matrix) = as.character(1:200)
colnames(expr_matrix) = paste("feature", 1:10)

feature_stability_result = get_feature_stability(data_matrix = t(expr_matrix),
  feature_set = colnames(expr_matrix),
  feature_type = "feature_name",
  steps = 5,
  npcs = 2,
  n_repetitions = 10,
```

```

algorithm = 1,
# the following parameters are used by the umap function and are not mandatory
n_neighbors = 3,
approx_pow = TRUE,
n_epochs = 0,
init = "random",
min_dist = 0.3)
plot_feature_stability_boxplot(feature_stability_result)

```

---

plot\_feature\_stability\_ecs\_facet

*Feature Stability - EC Consistency Facet Plot*

---

### Description

Display a facet of plots where each subpanel is associated with a feature set and illustrates the distribution of the EC consistency score over the UMAP embedding.

### Usage

```

plot_feature_stability_ecs_facet(
  feature_object_list,
  n_facet_cols = 3,
  point_size = 0.3
)

```

### Arguments

feature_object_list	An object or a concatenation of objects returned by the ‘get_feature_stability’ method
n_facet_cols	The number of facet’s columns.
point_size	The size of the points displayed on the plot.

### Value

A ggplot2 object

### Examples

```

set.seed(2021)
# create an artificial expression matrix
expr_matrix = matrix(c(runif(100*10), runif(50*10, min = 5, max = 7)), nrow = 150, byrow = TRUE)
rownames(expr_matrix) = as.character(1:150)
colnames(expr_matrix) = paste("feature", 1:10)

feature_stability_result = get_feature_stability(data_matrix = t(expr_matrix),
  feature_set = colnames(expr_matrix),

```

```
feature_type = "feature_name",
steps = c(5,10),
npcs = 2,
n_repetitions = 3,
algorithm = 1,
# the following parameters are used by the umap function and are not mandatory
n_neighbors = 3,
approx_pow = TRUE,
n_epochs = 0,
init = "random",
min_dist = 0.3)
plot_feature_stability_ecs_facet(feature_stability_result)
```

---

plot\_feature\_stability\_ecs\_incremental

*Feature Stability Incremental Boxplot*

---

## Description

Perform an incremental ECS between two consecutive feature steps.

## Usage

```
plot_feature_stability_ecs_incremental(
  feature_object_list,
  dodge_width = 0.7,
  text_size = 4,
  boxplot_width = 0.4
)
```

## Arguments

feature_object_list	An object or a concatenation of objects returned by the ‘get_feature_stability’ method.
dodge_width	Used for adjusting the horizontal position of the boxplot; the value will be passed to the ‘width’ argument of the ‘ggplot2::position_dodge’ method.
text_size	The size of the labels above boxplots.
boxplot_width	Used for adjusting the width of the boxplots; the value will be passed to the ‘width’ argument of the ‘ggplot2::geom_boxplot’ method.

## Value

A ggplot2 object with ECS distribution will be displayed as a boxplot. Above each boxplot there will be a pair of numbers representing the two steps that are compared.

**Examples**

```

set.seed(2021)
# create an artificial expression matrix
expr_matrix = matrix(c(runif(25*10), runif(75*10, min = 5, max = 7)), nrow = 100, byrow = TRUE)
rownames(expr_matrix) = as.character(1:100)
colnames(expr_matrix) = paste("feature", 1:10)

feature_stability_result = get_feature_stability(data_matrix = t(expr_matrix),
  feature_set = colnames(expr_matrix),
  feature_type = "feature_name",
  steps = c(5,10),
  npcs = 2,
  n_repetitions = 3,
  algorithm = 1,
  # the following parameters are used by the umap function and are not mandatory
  n_neighbors = 3,
  approx_pow = TRUE,
  n_epochs = 0,
  init = "random",
  min_dist = 0.3)
plot_feature_stability_ecs_incremental(feature_stability_result)

```

---

```
plot_feature_stability_mb_facet
```

*Feature Stability - Cluster Membership Facet Plot*

---

**Description**

Display a facet of plots where each subpanel is associated with a feature set and illustrates the distribution of the most frequent partition over the UMAP embedding.

**Usage**

```

plot_feature_stability_mb_facet(
  feature_object_list,
  text_size = 5,
  n_facet_cols = 3,
  point_size = 0.3
)

```

**Arguments**

feature_object_list	An object or a concatenation of objects returned by the ‘get_feature_stability’ method
text_size	The size of the cluster label
n_facet_cols	The number of facet’s columns.
point_size	The size of the points displayed on the plot.

**Value**

A ggplot2 object.

**Examples**

```
set.seed(2021)
# create an artificial expression matrix
expr_matrix = matrix(c(runif(100*10), runif(50*10, min = 5, max = 7)), nrow = 150, byrow = TRUE)
rownames(expr_matrix) = as.character(1:150)
colnames(expr_matrix) = paste("feature", 1:10)

feature_stability_result = get_feature_stability(data_matrix = t(expr_matrix),
  feature_set = colnames(expr_matrix),
  feature_type = "feature_name",
  steps = c(5,10),
  npcs = 2,
  n_repetitions = 3,
  algorithm = 1,
  # the following parameters are used by the umap function and are not mandatory
  n_neighbors = 3,
  approx_pow = TRUE,
  n_epochs = 0,
  init = "random",
  min_dist = 0.3)
plot_feature_stability_mb_facet(feature_stability_result)
```

---

plot_k_n_partitions	<i>Relationship Between the Number of Clusters and the Number of Unique Partitions</i>
---------------------	--

---

**Description**

For each configuration provided in `partition_obj_list`, display how many different partitions with the same number of clusters can be obtained by changing the seed.

**Usage**

```
plot_k_n_partitions(
  partition_obj_list,
  object_names = NULL,
  pt_size_range = c(1.5, 4),
  colour_information = c("ecc", "frequency_partition")
)
```

**Arguments**

`partition_obj_list`

An object or a concatenation of objects returned by the ‘merge\_resolutions’ method.



`object_names` Custom names that the user could assign to each configuration; if not specified, the plot will use the generated configuration names.

`pt_size_range` Indicates the minimum and the maximum size a point on the plot can have.

`colour_information` String that specifies the information type that will be illustrated using gradient colour: either 'frequency\_partition' for the frequency of the most common partition or 'ecc' for the Element-Centric Consistency of the partitions obtained when the number of clusters is fixed.

### Value

A ggplot2 object. The color gradient suggests the frequency of the most common partition relative to the total number of appearances of that specific number of clusters or the Element-Centric Consistency of the partitions. The size illustrates the frequency of the partitions with \*k\* clusters relative to the total number of partitions.

### Examples

```
set.seed(2021)
# create an artificial expression matrix
expr_matrix = matrix(runif(500*10), nrow = 500)

# get the PCA embedding of the data
pca_embedding = irlba::irlba(expr_matrix, nv = 2)
pca_embedding = pca_embedding$u %*% diag(pca_embedding$d)
rownames(pca_embedding) = as.character(1:500)

# run the function on the pca embedding
resolution_result = get_resolution_importance(embedding = pca_embedding,
  resolution = c(0.8, 1),
  n_neigh = c(5, 7),
  n_repetitions = 5,
  clustering_method = 1,
  graph_type = 2,
  object_name = "name_example")

plot_k_n_partitions(resolution_result)
```

---

plot\_k\_resolution\_corresp

*Correspondence Between Resolution and the Number of Clusters*

---

### Description

For each configuration provided in the `res_object_list`, display what number of clusters appear for different values of the resolution parameters.

**Usage**

```
plot_k_resolution_corresp(
  res_object_list,
  res_object_names = NULL,
  colour_information = c("ecc", "freq_k"),
  given_height = 0.7,
  pt_size_range = c(1.5, 4)
)
```

**Arguments**

**res\_object\_list** An object returned by the 'get\_resolution\_importance' method.

**res\_object\_names** Custom names that the user could assign to each configuration; if not specified, the plot will use the generated configuration names.

**colour\_information** String that specifies the information type that will be illustrated using gradient colour: either 'freq\_k' for the frequency of the number of clusters or 'ecc' for the Element-Centric Consistency of the partitions obtained when the resolution is fixed.

**given\_height** Used for adjusting the vertical position of the boxplot; the value will be passed in the 'width' argument of the 'ggplot::position\_dodge' method.

**pt\_size\_range** Indicates the minimum and the maximum size a point on the plot can have.

**Value**

A ggplot2 object. Different shapes of points indicate different parameter configuration, while the color illustrates the frequency of the most common partition or the Element-Centric Consistency of the partitions. The frequency is calculated as the fraction between the number of total appearances of partitions with a specific number of clusters and resolution value and the number of runs. The size illustrates the frequency of the most common partition with *k* clusters relative to the partitions obtained with the same resolution value and have *k* clusters.

**Examples**

```
set.seed(2021)
# create an artificial expression matrix
expr_matrix = matrix(runif(500*10), nrow = 500)

# get the PCA embedding of the data
pca_embedding = irlba::irlba(expr_matrix, nv = 2)
pca_embedding = pca_embedding$u %%% diag(pca_embedding$d)
rownames(pca_embedding) = as.character(1:500)

# run the function on the pca embedding
resolution_result = get_resolution_importance(embedding = pca_embedding,
  resolution = c(0.8, 1),
  n_neigh = c(5, 7),
```

```

n_repetitions = 5,
clustering_method = 1,
graph_type = 2,
object_name = "name_example")

plot_k_resolution_corresp(resolution_result)

```

---

plot\_n\_neigh\_ecs      *Graph construction parameters - ECC facet*

---

### Description

Display, for all configurations consisting in different number of neighbors, graph types and base embeddings, the EC Consistency of the partitions obtained over multiple runs on an UMAP embedding.

### Usage

```
plot_n_neigh_ecs(nn_ecs_object, boxplot_width = 0.5)
```

### Arguments

`nn_ecs_object` An object or a concatenation of objects returned by the ‘get\_nn\_importance’ method.

`boxplot_width` Used for adjusting the width of the boxplots; the value will be passed to the ‘width’ argument of the ‘ggplot2::geom\_boxplot’ method.

### Value

A ggplot2 object.

### Examples

```

set.seed(2021)
# create an artificial expression matrix
expr_matrix = matrix(c(runif(100*10), runif(100*10, min=5, max=6)), nrow = 200)
rownames(expr_matrix) = as.character(1:200)

nn_importance_obj = get_nn_importance(object = expr_matrix,
  n_neigh_sequence = c(10,15,20),
  n_repetitions = 10,
  graph_reduction_type = "PCA",
  algorithm = 1,
  transpose = FALSE, # the matrix is already observations x features, so we won't transpose it
  # the following parameter is used by the irlba function and is not mandatory
  nv = 2)
plot_n_neigh_ecs(nn_importance_obj)

```

---

plot\_n\_neigh\_k\_correspondence

*Relationship Between Number of Nearest Neighbors and Number of Clusters*

---

### Description

Display the distribution of the number of clusters obtained for each number of neighbors across random seeds.

### Usage

```
plot_n_neigh_k_correspondence(nn_object_n_clusters)
```

### Arguments

nn\_object\_n\_clusters

An object or a concatenation of objects returned by the ‘get\_nn\_importance’ method.

### Value

A ggplot2 object with the distributions displayed as boxplots.

### Note

The number of clusters is displayed on a logarithmic scale.

### Examples

```
set.seed(2021)
# create an artificial expression matrix
expr_matrix = matrix(runif(100*10), nrow = 100)
rownames(expr_matrix) = as.character(1:100)

nn_importance_obj = get_nn_importance(object = expr_matrix,
  n_neigh_sequence = c(2,5),
  n_repetitions = 5,
  graph_reduction_type = "PCA",
  algorithm = 1,
  transpose = FALSE, # the matrix is already observations x features, so we won't transpose it
  # the following parameter is used by the irlba function and is not mandatory
  nv = 2)
plot_n_neigh_k_correspondence(nn_importance_obj)
```

# Index

consensus\_cluster, 2

element\_agreement, 4  
element\_consistency, 5  
element\_sim, 7  
element\_sim\_elscore, 9  
element\_sim\_matrix, 11

get\_clustering\_difference, 12  
get\_feature\_stability, 14  
get\_nn\_conn\_comps, 15  
get\_nn\_importance, 17  
get\_resolution\_importance, 19

marker\_overlap, 21  
merge\_partitions, 22

pac\_convergence, 23  
pac\_landscape, 24  
plot\_clustering\_difference\_boxplot, 24  
plot\_clustering\_difference\_facet, 25  
plot\_connected\_comps\_evolution, 27  
plot\_feature\_stability\_boxplot, 28  
plot\_feature\_stability\_ecs\_facet, 29  
plot\_feature\_stability\_ecs\_incremental,  
30  
plot\_feature\_stability\_mb\_facet, 31  
plot\_k\_n\_partitions, 32  
plot\_k\_resolution\_corresp, 33  
plot\_n\_neigh\_ecs, 35  
plot\_n\_neigh\_k\_correspondence, 36