Package ‘ClustAssess’

March 31, 2021

Type Package

Title Tools for Assessing Clustering

Version 0.1.1

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

Description A set of tools for evaluating clustering similarity across methods and method stability using element-centric clustering comparison (Gates et al. (2019) <doi:10.1038/s41598-019-44892-y>). Additionally, this package enables data-wide assessment of clustering robustness using proportion of ambiguously clustered pairs (Senbabaoglu et al. (2014) <doi:10.1038/srep06207>), which can be used to infer the optimal number of clusters in the data.

License MIT + file LICENSE

Encoding UTF-8

Imports ggplot2, dplyr, fastcluster, rlang, Matrix, igraph, magrittr, Rcpp, methods, stats

RoxygenNote 7.1.1

LinkingTo Rcpp

Suggests knitr, rmarkdown, e1071, dbscan, dendextend, Seurat

URL https://github.com/Core-Bioinformatics/ClustAssess

VignetteBuilder knitr

NeedsCompilation yes

Author Arash Shahsavari [aut, cre], Irina Mohorianu [aut]

Repository CRAN

Date/Publication 2021-03-31 16:40:03 UTC

R topics documented:

Clustering-class .................................................. 2
consensus_cluster .............................................. 3
Clustering-class

The Clustering Class

Description
A class containing relevant data for comparing clusterings, including the affinity matrix for the Clustering.

Slots
names A character vector of element names; will be 1:n_elements if no names were available when creating the Clustering object.
n_elements A numeric giving the number of elements.
is_hierarchical A logical indicating whether the clustering is hierarchical or flat.
is_disjoint A logical indicating whether the clustering is disjoint or overlapping.
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.
elm2clu_dict A list giving the clusters each element is a member of.
clu2elm_dict A list giving the element members of each cluster.
affinity_matrix A Matrix containing the personalized pagerank equilibrium distribution.

Examples
km.res = kmeans(mtcars, 3)$cluster
km.clustering = create_clustering(km.res)
hc.res = hclust(dist(mtcars))
hc.clustering = create_clustering(hc.res)
element_sim(km.clustering, hc.clustering)
Description

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

Usage

```r
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
)
```

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 `pac_convergence`.
- `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 `hclust`; must be one of "euclidean", "maximum", "manhattan", "canberra", "binary" or "minkowski".
- `linkage`: The linkage method used in `hclust`; 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.
create_clustering

Value

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

References


Examples

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

create_clustering Create Clustering Object

Description

Creates a Clustering object from the output of a clustering method.

Usage

```r
create_clustering(clustering_result, ...)  
## S4 method for signature 'numeric'
create_clustering(clustering_result, alpha = 0.9)  
## S4 method for signature 'character'
create_clustering(clustering_result, alpha = 0.9)  
## S4 method for signature 'factor'
create_clustering(clustering_result, alpha = 0.9)  
## S4 method for signature 'matrix'
create_clustering(  
  clustering_result,  
  alpha = 0.9,  
  ppr_implementation = "prpack",  
  row_normalize = TRUE
)
## S4 method for signature 'Matrix'
create_clustering(
```
create_clustering

clustering_result,
alpha = 0.9,
ppr_implementation = "prpack",
row_normalize = TRUE )

## S4 method for signature 'hclust'
create_clustering(
clustering_result,
alpha = 0.9,
r = 1,
rescale_path_type = "max",
ppr_implementation = "prpack",
dist_rescaled = FALSE )

Arguments

clustering_result
The clustering result, 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.

ppr_implementation
Choose a implementation for personalized page-rank calculation:
  • 'prpack': use PPR algorithms in igraph.
  • 'power_iteration': use power_iteration method.

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.

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.

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

Value

A Clustering object.
Methods (by class)

- numeric: Create Clustering Object from Numeric Vector
- character: Create Clustering Object from Character Vector
- factor: Create Clustering Object from Factor Vector
- matrix: Create Clustering Object from base matrix
- Matrix: Create Clustering Object from Matrix::Matrix
- hclust: Create Clustering Object from hclust

Examples

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

---

**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

```r
element_agreement(reference_clustering, clustering_list)
```

Arguments

- `reference_clustering`:
  A Clustering objects for the reference clustering that each clustering in clustering_list is compared to.

- `clustering_list`:
  A list of Clustering objects used to calculate the element-wise average agreement

Value

A vector containing the element-wise average agreement.

References

element_frustration

Examples

reference.clustering = create_clustering(iris$Species)
clustering.list = list()
for (i in 1:20){
  km.res = kmeans(iris[,1:4], 3)$cluster
  clustering.list[[i]] = create_clustering(km.res)
}
element_agreement(reference.clustering, clustering.list)

element_frustration Element-Wise Frustration Between a Set of Clusterings

Description

Inspect the consistency of a set of clusterings by calculating their element-wise clustering frustration.

Usage

element_frustration(clustering_list)

Arguments

clustering_list

A list of Clustering objects used to calculate the element-wise frustration.

Value

a vector containing the element-wise frustration.

References


Examples

clustering.list = list()
for (i in 1:20){
  km.res = kmeans(mtcars, 3)$cluster
  clustering.list[[i]] = create_clustering(km.res)
}
element_frustration(clustering.list)
**element_sim**  
*The Element-Centric Clustering Similarity*

**Description**
Calculates the average element-centric similarity between two Clustering objects.

**Usage**
```
element_sim(clustering1, clustering2)
```

**Arguments**
- `clustering1` The first Clustering.
- `clustering2` The second Clustering.

**Value**
The average element-wise similarity between the two Clusterings.

**Examples**
```r
km.res = kmeans(mtcars, 3)$cluster
c.create_clustering(km.res)
hc.res = hclust(dist(mtcars))
hc.clustering = create_clustering(hc.res)
element_sim(km.clustering, hc.clustering)
```

**element_sim_elscore**  
*The Element-Centric Clustering Similarity for each Element*

**Description**
Calculates the element-wise element-centric similarity between two Clustering objects.

**Usage**
```
element_sim_elscore(clustering1, clustering2)
```

**Arguments**
- `clustering1` The first Clustering.
- `clustering2` The second Clustering.

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


Examples

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

---

**element_sim_matrix**

**Pairwise Comparison of Clusterings**

**Description**

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

**Usage**

```r
element_sim_matrix(clustering_list, output_type = "matrix")
```

**Arguments**

- `clustering_list` - A list of Clustering objects to be compared with element-centric similarity.
- `output_type` - A string specifying whether the output should be a matrix or a data.frame.

**Value**

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

**References**


**Examples**

```r
clustering.list = list()
for (i in 1:20){
  km.res = kmeans(mtcars, 3)$cluster
  clustering.list[[i]] = create_clustering(km.res)
}
element_sim_matrix(clustering.list, output_type='matrix')
```
length, Clustering-method

Length of an Object

Description
Get the number of elements in the Clustering.

Usage
```r
## S4 method for signature 'Clustering'
length(x)
```

Arguments
- `x`: The Clustering object.

Value
The number of elements.

Examples
```r
km.res = kmeans(mtcars, 3)$cluster
kmc.clustering = create_clustering(km.res)
length(kmc.clustering)
```

marker_overlap

Cell-wise marker gene overlap

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

Usage
```r
marker_overlap(
  markers1,
  markers2,
  clustering1,
  clustering2,
  n = 25,
  overlap_type = "jsi",
  rank_by = "-p_val"
)
```
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.

Value

A vector of the marker gene overlap per cell.

Examples

```r
suppressWarnings({
  set.seed(12345)
  library(Seurat)

  # 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$kmeans_clusters = kmeans(pbmc.pca, centers=2)$cluster

  # compare the markers
  Idents(pbmc_small) = pbmc_small@meta.data$seurat_clusters
  louvain.markers = FindAllMarkers(pbmc_small, logfc.threshold=1, verbose=FALSE)
  Idents(pbmc_small) = pbmc_small@meta.data$kmeans_clusters
  kmeans.markers = FindAllMarkers(pbmc_small, logfc.threshold=1, 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")
})
```
pac_convergence  

**PAC Convergence Plot**

**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.

**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 number of iterations to shade to show the variability of PAC across the last n_shade iterations.

**Value**
A ggplot2 object with the final PAC vs k plot.
Examples

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

Description

Prints out information about the Clustering, including number of elements.

Usage

```r
## S4 method for signature 'Clustering'
print(x)
```

Arguments

- `x` The Clustering object.

Value

The printed character string.

Examples

```r
km.res = kmeans(mtcars, 3)$cluster
km.clustering = create_clustering(km.res)
print(km.clustering)
```
Index

Clustering (Clustering-class), 2
Clustering-class, 2
consensus_cluster, 3
create_clustering, 4
create_clustering, character-method
  (create_clustering), 4
create_clustering, factor-method
  (create_clustering), 4
create_clustering, hclust-method
  (create_clustering), 4
create_clustering, Matrix-method
  (create_clustering), 4
create_clustering, matrix-method
  (create_clustering), 4
create_clustering, numeric-method
  (create_clustering), 4

element_agreement, 6
element_frustration, 7
element_sim, 8
element_sim_elscore, 8
element_sim_matrix, 9

length, Clustering-method, 10

marker_overlap, 10

pac_convergence, 12
pac_landscape, 12
print, Clustering-method, 13